

# **PDP-10 TIMESHARING HANDBOOK**

**Prepared by  
The PDP-10 Software Writing Group  
Programming Department  
Digital Equipment Corporation**

Additional copies of this handbook may be ordered from the Program Library, Digital Equipment Corporation, Maynard, Mass. 01754. Order code AKW. \$5.00 each. Discounts are available on five or more copies.

~~100-1000000~~ 301

001.642  
D569pt  
1970

**PDP-10 HANDBOOK SERIES**

**All rights reserved.**

**Permission to reproduce this handbook or any parts thereof may be obtained from the PDP-10 Product Line Manager, Digital Equipment Corporation, Maynard, Mass.**

**The material in this handbook is for information purposes and is subject to change without notice.**

**Copyright © 1968, 1969, 1970 by  
Digital Equipment Corporation**

**The following are trademarks of Digital Equipment Corporation,  
Maynard, Massachusetts:**

**DEC**

**Digital**

**PDP**



**INTRODUCTION TO TIMESHARING**

(pages 1-1 thru 1-12)

**GETTING STARTED WITH THE MONITOR**

(pages 2-1 thru 2-20)

**BASIC**

(pages 3-1 thru 3-78)

**AID**

(pages 4-1 thru 4-99)

**FORTRAN**

(pages 5-1 thru 5-133)

**DEMONSTRATION PROGRAMS**

(pages 6-1 thru 6-49)

**ADVANCED MONITOR COMMANDS**

(pages 7-1 thru 7-21)

**UTILITY PROGRAMS**

(pages 8-1 thru 8-89)

**APPENDICES**

(pages A-1 thru B-4)

**INDEX**

(pages Index-1 thru Index-10)





### FOREWORD

We have written this handbook for the individual with little or no programming skill in an attempt to bring timesharing programming competence to an ever-expanding circle of new computer users. With this volume as his guide, we hope he can soon acquire the necessary programming knowledge to improve his business or professional activity by the application of computer technology.

I'm pleased to acknowledge here the work of the many DEC programmers, designers, and engineers who continue to advance the state of the timesharing art in both hardware and software, and the DEC software writers and technical artists who prepared this volume.



President, Digital Equipment Corporation

## PREFACE

In developing its timesharing capability, Digital has built a history of success very similar to the company's record in realtime applications. That history started in 1960 when Digital's customers began building timesharing systems around PDP computers. Three years later Digital itself started development of its own timesharing system, the PDP-6; and in 1964 the PDP-6 became the first timesharing computer to be delivered with manufacturer-supplied hardware and software.

The PDP-10, which emerged in 1967, is the successful culmination of many years of computer research. Its power, versatility, and low cost make it a leader in the general-purpose timesharing field. For its timesharing users, the PDP-10 performs scientific data analyses, helps make better management decisions, aids in engineering and architectural design, makes investment analyses, and provides management information services.

With this handbook, Digital attempts to bring its documentation on timesharing to a par with its hardware and software accomplishments. The handbook is intended primarily for students, scientists, engineers, and financial analysts who have little or no experience in programming. From it they can learn timesharing programming from a remote Teletype using disk input/output.

This is not to say that an experienced programmer is automatically debarred from using this document. If the reader happens to be a programmer, he should skip the preliminary books, go straight to the computer language in Book 5, and commence programming. In Book 6 he will find that Demonstration Programs 3 and 4 are geared to his level of programming knowledge and competence.

A synoptic view of the contents of the handbook is as follows. Book 1 describes the evolutionary history of timesharing and gives the reader an insight into the way it operates. Book 2, in explaining the elementary monitor commands, shows the reader how to get on the system. In Books 3 and 4 the reader will find conversational programming with BASIC and AID, respectively. Book 5, as already indicated, contains FORTRAN. Four demonstration programs constitute Book 6; advanced monitor commands are found in Book 7; and the four utility programs Batch, CHAIN, LINED, and TECO appear in Book 8.

Since the handbook will be revised periodically in order to improve it and keep it up to date, we solicit the reader's constructive evaluations in the questionnaire at the back of the book. Please fill out the questionnaire and return it to

PDP-10 Software Writing Group  
Programming Department  
Digital Equipment Corporation  
Maynard, Massachusetts 01754

A companion volume, the PDP-10 Reference Handbook, is likewise in print. It is oriented toward experienced programmers who are interested in writing and operating assembly-language programs.



## CONTENTS

|                                                                                                                                                                                                                                                                                                                                                   |         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| Foreword .....                                                                                                                                                                                                                                                                                                                                    | V       |
| Preface .....                                                                                                                                                                                                                                                                                                                                     | VI      |
| Book 1 Introduction to Timesharing .....                                                                                                                                                                                                                                                                                                          | 1-1     |
| A general description of the operation of a variety of factors in the evolution of timesharing.                                                                                                                                                                                                                                                   |         |
| Book 2 Getting Started with the Monitor .....                                                                                                                                                                                                                                                                                                     | 2-1     |
| Logging in, description of files, elementary commands to create, edit, manipulate, translate, load, and execute files, getting information from the system, and logging off the system.                                                                                                                                                           |         |
| Book 3 Conversational Programming with BASIC .....                                                                                                                                                                                                                                                                                                | 3-1     |
| A complete book, explaining the procedures for logging in, logging off, and writing, editing, and running programs in BASIC—a problem-solving conversational language that can be used to solve both simple and complex mathematical problems.                                                                                                    |         |
| Book 4 Conversational Programming with AID .....                                                                                                                                                                                                                                                                                                  | 4-1     |
| A complete book explaining how to program in AID, an algebraic conversational language designed to solve both simple and complex numerical problems.                                                                                                                                                                                              |         |
| Book 5 Programming in FORTRAN .....                                                                                                                                                                                                                                                                                                               | 5-1     |
| A reference book describing the specific statements and features of the FORTRAN IV language, designed primarily for the experienced FORTRAN programmer.                                                                                                                                                                                           |         |
| Book 6 Demonstration Programs .....                                                                                                                                                                                                                                                                                                               | 6-1     |
| Four programs designed to utilize concepts and commands discussed in this manual. Demonstrations three and four also make use of advanced commands in TECO and DDT.                                                                                                                                                                               |         |
| Book 7 Advanced Monitor Commands .....                                                                                                                                                                                                                                                                                                            | 7-1     |
| Explanation of monitor commands not covered in Book 2—commands to allocate system resources, produce line printer listings, manipulate core images, start a program, and get information from the system.                                                                                                                                         |         |
| Book 8 Utility Programs .....                                                                                                                                                                                                                                                                                                                     | 8-1     |
| Includes BATCH—sequential execution of a series of jobs, CHAIN—allows users to deal with FORTRAN programs too large to fit or load into the amount of core available, LINED—a line-oriented editor designed for use on the disk, and TECO—description and explanation of the most frequently used commands of this powerful text editing program. |         |
| Appendices .....                                                                                                                                                                                                                                                                                                                                  | A-1     |
| Master Index .....                                                                                                                                                                                                                                                                                                                                | Index-1 |





# **Book 1**

## **Introduction to Timesharing**

# INTRODUCTION TO JOURNALISM

## Book 1

## Introduction

## Journalism

mode of operation. The journalist's way to interact with the information



## INTRODUCTION TO TIMESHARING

### Why Timesharing?

Early computers were the province of the mathematician. Used mainly to solve differential equations, the systems were narrow in scope and poorly utilized. Since few persons were knowledgeable enough to employ the enormous processors, one individual could monopolize computer time—sit at the console and solve problems in step-by-step fashion.

As more people discovered computing techniques, it was no longer practical to let a few persons monopolize computer time. To increase machine efficiency, batch processing was introduced. In this mode of operation, no time was wasted between jobs. Programs were punched on cards and the cards stacked and fed to the computer in batches. Operation of each program was governed by control cards that took the place of the human operator.

### batch processing

Since card reading is a relatively slow process, some early systems employed a small computer to read the cards and transfer program information to magnetic tape that was then input to the large computer. As a further refinement, programs were assigned priorities, with short jobs being executed first to minimize job turnaround.

But what about the computer user? As computer utilization improved, program development took more time. To develop a new program, a user performed the following procedure. After writing the program on paper, he carried it to a keypunch operator to have the cards punched and verified. A day or so later, when the program was returned, the user checked for punching errors, then returned to the keypunch for corrections.

Next, he sent the cards to the computer center for compilation. The compilation, which might not be returned for a half day or more, could reveal spelling or syntactical errors. The cards then had to be changed and resubmitted—another half day's wait. If the next compilation was successful and the program was run, program logic errors might be discovered—new cards, new compilation, etc., etc. In addition, the user often studied reams of computer listings to find the errors. Using these inefficient methods, even simple programs might take weeks to develop.

Batch processing maximizes machine efficiency in routine data processing operations where turnaround is not critical. But for program development and modification, the user requires another mode of operation. The user needs a way to "interact" with the computer—to feed his program to the system, line by line, and continuously check the results.

### interaction

In fact, the user may want to develop interactive programs. These programs, which are extremely productive tools, ask the user questions and perform an analysis based on his answers. Electronic circuit design programs are a prime example. The computer actually designs the circuit by asking the engineer questions and manipulating his answers. In addition, interaction provides a new dimension in management information reporting. Via an interactive terminal, a manager can request summaries, plot trends in plant operation and sales, and select special data for use in decision making.

#### **dedicated system**

If the user had unlimited funds, he might be tempted to buy or lease a large computer—a system he could dedicate to his work that would provide sufficient power, many peripherals, and a large variety of software. With such a system, the user could develop programs interactively or utilize batch processing for routine tasks. However, costs in excess of \$20,000 per month normally preclude the dedication of a large system to a single user.

#### **timesharing**

By using timesharing, the user has most of the benefits of a dedicated system at a small fraction of the cost. Timesharing with today's technology allows a large powerful computer to handle 20, 50, 100 or more users simultaneously. Through a choice of terminals, the user can interact with the system or initiate batch processing which runs concurrently. The user also has access to a choice of mass storage and peripherals and a selection of languages and application programs. Since response is fast, the user appears to have a dedicated system. Yet costs are shared. He pays only for the time and facilities that he requires and doesn't pay for the time the machine is idle.

### **The Operation of a Timesharing System**

A timesharing system isn't just any computer with some additional hardware and software. It's a system designed specifically for timesharing. Otherwise, facilities are limited, fewer users can be handled efficiently, and economics are unattractive. At a minimum, a timesharing system requires a central processor with sufficient speed and power, input/output terminals, and an amount of core memory adequate to hold several users.

#### **time slice time quantum round robin operation**

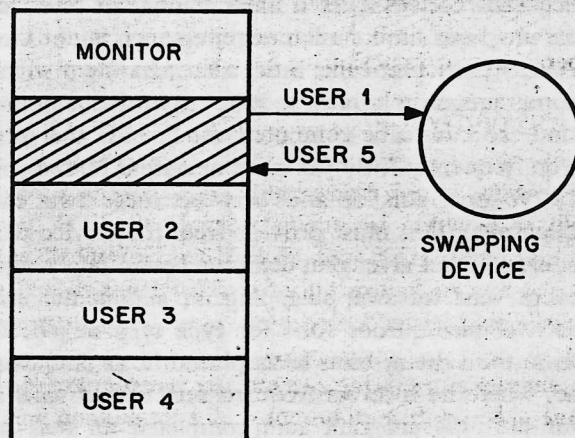
In a simple timesharing system, each program is assigned a fixed time slice or time quantum and operation is switched from one program to another in round robin fashion until each program is completed. Essentially, if each user receives 1/60 of a second and 12 users are "on" the system, each user will receive service every 1/5 of a second.



The timesharing system performs multiprogramming; that is, it allows several programs to reside in core simultaneously and to operate sequentially. The switching between programs, called context switching, is initiated by a clock which interrupts the central processor to signal that a certain time period has elapsed. The interrupt function is provided by a priority interrupt system. A monitor, also called an operating system or executive program, directs the execution of these tasks and performs other housekeeping duties.

The monitor is also involved in keeping the actions of a user within his assigned memory space. A hardware device, a memory protection register, which is set by the monitor, limits the core area that a particular user can access. Any attempt by the program to read or change information outside that limit will automatically stop the program and notify the monitor.

The system discussed so far services a number of users sequentially in round robin fashion. To increase the number of users serviced, more main memory or core is required. However, since core is expensive, a secondary memory is employed. This memory—usually magnetic disk or drum—is slower than core or main memory but provides greatly increased capacity at reasonable cost. User programs can be located in secondary memory and moved into main memory for execution. Programs entering main memory exchange places with a program (or programs) that has just been serviced by the central processor. This operation is called swapping (see diagram).



**SWAPPING**

**multiprogramming**

**context switching , clock**

**priority interrupt system  
monitor , operating system  
executive program**

**memory protection register**

**main memory  
secondary memory**

**swapping**

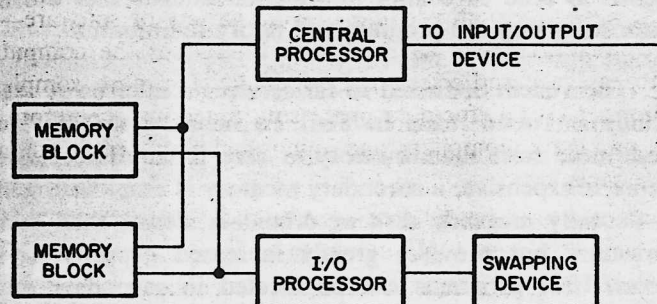
**memory blocks**

**input/output processor**

**asynchronous design**

In operation, main memory is divided into separate memory blocks. Secondary memory is connected to these blocks through a high speed input/output processor—a hardware device that allows the disk or drum to swap a program into any one of the main memory blocks without any aid from the central processor. This structure allows the central processor to be operating a user program in one block of memory while programs are being swapped to and from another block. This independent overlapped operation, which greatly improves efficiency and processing power, is characteristic of an asynchronous system design philosophy. See diagram.

#### MEMORY STRUCTURE



#### Dynamic Scheduling

Round robin scheduling, in which each program operates in sequence and receives a fixed amount of time, is effective only if all programs have similar requirements. Such is not the case, however. At any particular time, a timesharing system will be handling some programs which require extensive amounts of computing time (and are said to be compute bound) and other programs that must stop frequently for input or output (I/O bound).

To serve programs at and between these two extremes, the scheduling algorithm must provide frequent service to I/O bound programs and must give compute bound jobs longer time quantum to prevent wasteful swapping. A simple dynamic scheme could provide two queues—one for each type of job. When a user first logs on to the system, he is placed in an I/O bound queue (waiting line) where he receives frequent service and small time quantum. If the program isn't completed or does not request input or output during the time allotted to him, the job needs more computing time and is placed in the compute bound queue. Thus the scheduling algorithm optimizes system efficiency by automatically adjusting to program requirements.

**compute bound**

**I/O bound**

**scheduling algorithm**

**queue**



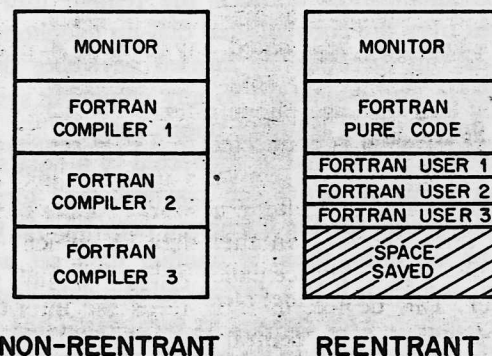
In the present state of scheduling art, algorithms are constantly being changed and improved. Current algorithms are extremely sophisticated, providing excellent service for most timesharing job mixes. They also allow fine tuning, if such modifications are necessary. The ability of the algorithm to match processing to program requirements insures the best service possible for all user programs.

In an efficient timesharing system, monitor functions (referred to as monitor overhead) take 5 to 10 percent of central processor time, making 90 to 95 percent of the time available to users.

**monitor overhead**

### Sharing Software

Since users of large timesharing systems have varying requirements, a good system provides a wide variety of software—interactive languages such as BASIC and AID for the computations of the engineer and scientist, FORTRAN for more complex calculations, COBOL for data processing functions. Therefore many users can have compilers and other common programs in core at the same time.



To prevent excessive core usage which results when a program is duplicated for several users, reentrant software is employed. That is, the program is written in two parts. One part contains pure code that is not modified during execution and can be used to simultaneously service any number of users. For example, the pure code portion of FORTRAN can service multiple FORTRAN users. A separate second part of the program belongs strictly to each user and consists of the code and data that is developed during the compiling process (impure code). This section is stored in a separate area of core. A comparison of memory usage in the non-reentrant and reentrant systems is shown in the diagram above.

**reentrant software**

**pure code**

**impure code**

What are the benefits of reentrant software? First, less core is required. For example, a reentrant system can service three FORTRAN users with one 8K compiler and three 2K user areas, a total of 14K. A non-reentrant system would require 30K for the three 8K compilers and three 2K user areas. Total savings in this case is 16K of core. Using less core means that more programs can fit into a given amount of space. The monitor then swaps less often and spends less time swapping the smaller impure sections.

There are other savings too. Since the pure code never changes, it doesn't have to be returned to disk storage (swapped out). As long as a single copy is maintained on the disk, it can be called into core at any time. Programs can be swapped in or "overlayed" on top of the compiler to take its place in core whenever the compiler is not needed.

overlay

dual memory protection  
and relocation

To protect the pure code from being modified, a hardware feature is provided—dual memory protection and relocation. This feature allows a program to execute as two separate segments, one of which is protected. User programs can also be written to make use of this protection. For example, a user might develop a reentrant information retrieval system written in COBOL.

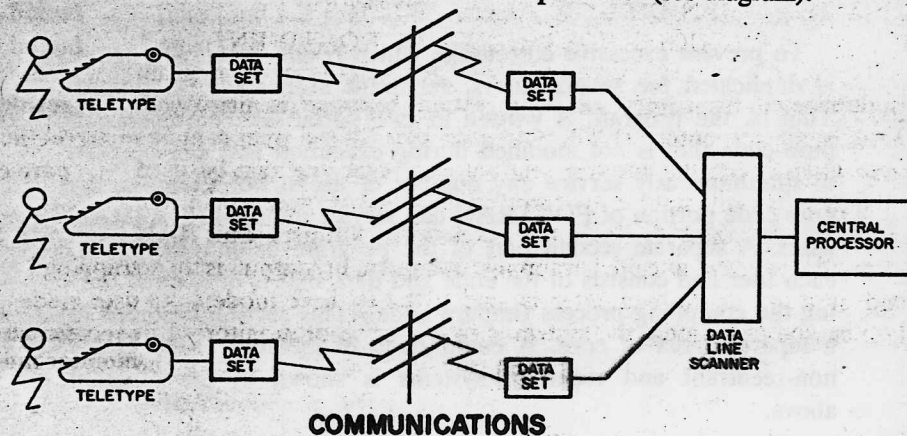
### Communications

Communication between the remote user and the computer passes over the conventional dial-up telephone network. User terminals can therefore be located anywhere that phone service is available and connected to any computer system, feasibility limited only by long distance phone rates.

data set  
modem

data line multiplexor  
data line scanner

Each user terminal is connected to a data set or modem (modulator-demodulator) which converts user terminal output into a signal suitable for the telephone network. At the computer end of the phone lines, there is another data set which reconverts the signal and feeds it to a device called a data line multiplexor or data line scanner. This device, in turn, feeds the information from a number of terminals to the central processor (see diagram).





The number of data sets employed at the user end of the system is unlimited. At the computer end of the communications network, however, the number of data sets is limited by the number of users that can be serviced simultaneously by the system.

In order to gain access to the system, the user dials the system phone number from his data set. The telephone network handles the call, scanning the data sets at the computer system. If all of the sets are busy, the user receives a busy signal, just as he would with normal phone service. If a set is available, the telephone network rings it, causing the data line scanner to interrupt the monitor. The computer answers the call, placing the user in communication with the monitor. The terminal is then on-line and ready for operation. **on-line**

### **Control of Input/Output**

A timesharing system has performed its basic function if it allows a number of users simultaneous access to a central computer. However, to be fully useful, the system should also allow the users access to other system resources—storage devices for his programs and data, line printers, card readers, etc. For example, the user should be able to choose between magnetic tape and disk for program storage. And if he has a 50-page report to produce, he should be able to employ a line printer instead of his Teletype®. If users controlled these devices, however, much confusion might result. For example, two users might select the line printer at the same time. If one user was processing Abraham Lincoln's Gettysburg Address and another, Mark Anthony's funeral oration, the report might look like the following:

I COME TO BURY CAESAR NOT TO PRAISE HIM  
FOUR SCORE AND SEVEN YEARS AGO  
THE EVIL THAT MEN DO LIVE AFTER THEM  
OUR FATHERS BROUGHT FORTH ON THIS CONTINENT

To prevent users from interfering with each other, the monitor coordinates input and output (I/O). The processor has an operating mode switch which the monitor sets before a user program is run. If the program attempts to perform input or output, the user program is stopped and the monitor takes over. Control thus diverted to the monitor is called trapping. When input/output is prevented or trapped, the computer is said to be in user mode; when I/O can be performed, the system is in executive or monitor mode.

**input/output control**

**trapping**  
**user mode**  
**executive mode**  
**monitor mode**

®—registered trademark of Teletype Corporation, Skokie, Illinois

When the system is in user mode, the memory protection feature is in operation. In monitor mode, this feature is disabled and the monitor has access to all of core. User mode also prevents the user from issuing a HALT command, which could stop operation of the entire system.

User-to-monitor-mode switching occurs when the user requests I/O or other special functions to be performed by the monitor. The requests are made by using computer instructions referred to as monitor calls or programmed operators. For more information see PDP-10 REFERENCE HANDBOOK.

**monitor calls  
programmed operators**

Since I/O is handled by the monitor, input or output can be transferred even if the user program is not in main memory. The monitor can also optimize throughput, keeping all devices busy simultaneously (overlapping of I/O operations) and executing jobs in the most efficient order. For example, it will start the read mechanisms on several disk packs in motion, simultaneously, to reduce the time required to find the desired data on each pack (access time). In addition, by means of the disk pack controller, the monitor can determine which of all needed data on a pack is closest to the read mechanism and can be obtained in the shortest amount of time (latency optimization).

**overlapped I/O**

**access time**

**latency optimization**

### **File Handling**

If a user does not require a fast device for his exclusive use (private device), he can elect to use a public device, in effect performing timesharing with a disk or drum. Under these conditions, user programs and data coexist on the device. Therefore, a filing system is necessary if program and data segments are to be retrieved in proper order.

**private device  
public device**

**filing system**

Data is transferred from memory to a peripheral device as a block of words or a record. (A word is the number of binary digits or bits that the central processor can retrieve and "operate on" at one time.) Record length can be arbitrary or dictated by the physical device being used, for example, the number of columns on an 80 column card or on a 132 column line printer. For PDP-10 disk files, the length is 128 words, so that blocks of 128 words are written at one time on a disk or other similar device.

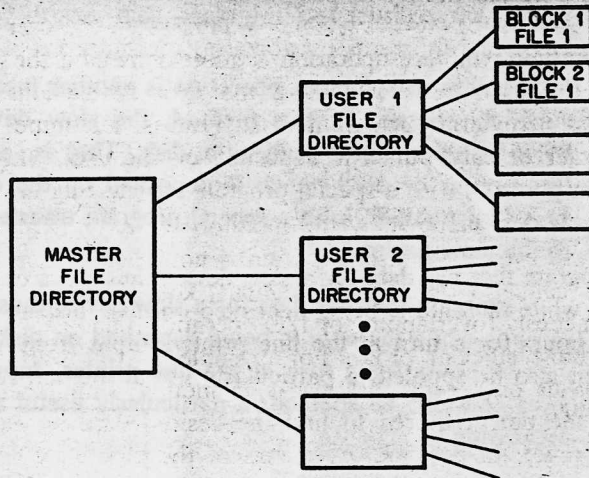
**block of words  
record**

For convenience each user's blocks are organized in groups called files which are listed in proper order in a special block on the disk called the User's File Directory (UFD). A Master File Directory (MFD) is then required to maintain the locations

**User's File Directory (UFD)  
Master File Directory (MFD)**



of the User's File Directories and also keep track of the number of blocks of free storage that can be assigned to new files. The resulting hierarchy is shown in the following diagram.



**FILE STRUCTURE**

Files, like memory, must be protected from access by unauthorized users. When a user closes a file, he can restrict it, specifying whether others can have access, and if access is permitted, whether the files can be modified or only read. With such an arrangement, programmers in various plant locations can use the same data to work simultaneously on the same project. But unauthorized personnel cannot modify or read the files.

#### **file protection**

#### **Slow Peripherals**

Fast peripherals can be timeshared. But what about the slow peripherals, such as the line printer and the card reader? Should other users be required to wait 20 minutes or so while one user ties up the line printer?

To eliminate conflicts, the user can request a slow device for his exclusive or private use. For example, he can request the line printer or card reader. Also available for private use are removable storage devices such as magnetic tape, DECTape, (DIGITAL'S low cost, high reliability magnetic tape), or disk packs. If the device is not already assigned to another user, the

#### **removable storage device**

**spooling  
symbiont operation**

monitor grants his request and the user has the device at his disposal until he releases it. For example, the user could request the use of multiple disk pack drives (exclusive use) to sort a payroll transaction file. Or he could assign himself a DECTape drive and ask the system operator to mount the DECTape that contains his own personal library of programs.

Spooling or symbiont operation is another method for handling slow peripherals. In this method, the slow device is simulated by a fast peripheral such as a disk. That is, all output for the line printer or card punch is deposited on the disk. The disk is later "unspooled", with a special program transferring information to the slow device.

A program that has data for a slow device thus waits only milliseconds while the data is being deposited on disk, instead of minutes or hours for a turn at the line printer. Input from slow devices can also be spooled, a particularly useful method for batch processing.

**Reliability**

With a large number of users depending on its operation, the timesharing system must be extremely reliable. A system with 99 percent reliability can be "down" 14 minutes during a 24-hour working day. If that 14 minutes affects only one user, reliability may be acceptable. However, if it affects a large number of users, the consequences are much more serious.

The problem is also complicated by the fact that reliability is a function of both hardware and software. It may take years, for example, to experience all the events that could uncover an error in software as complex as a timesharing monitor.

**modularity**

Today's hardware and software has reliability built in. Hardware is designed in modular fashion so that failed components can be removed and new replacements "plugged in". Some components also contain self-testing features that detect potential failures. Software is designed to be "defensive," that is, it anticipates certain types of failures and helps to minimize their effects. For example, the software might note parity errors and limit their effect to the program being operated.

**defensive software**

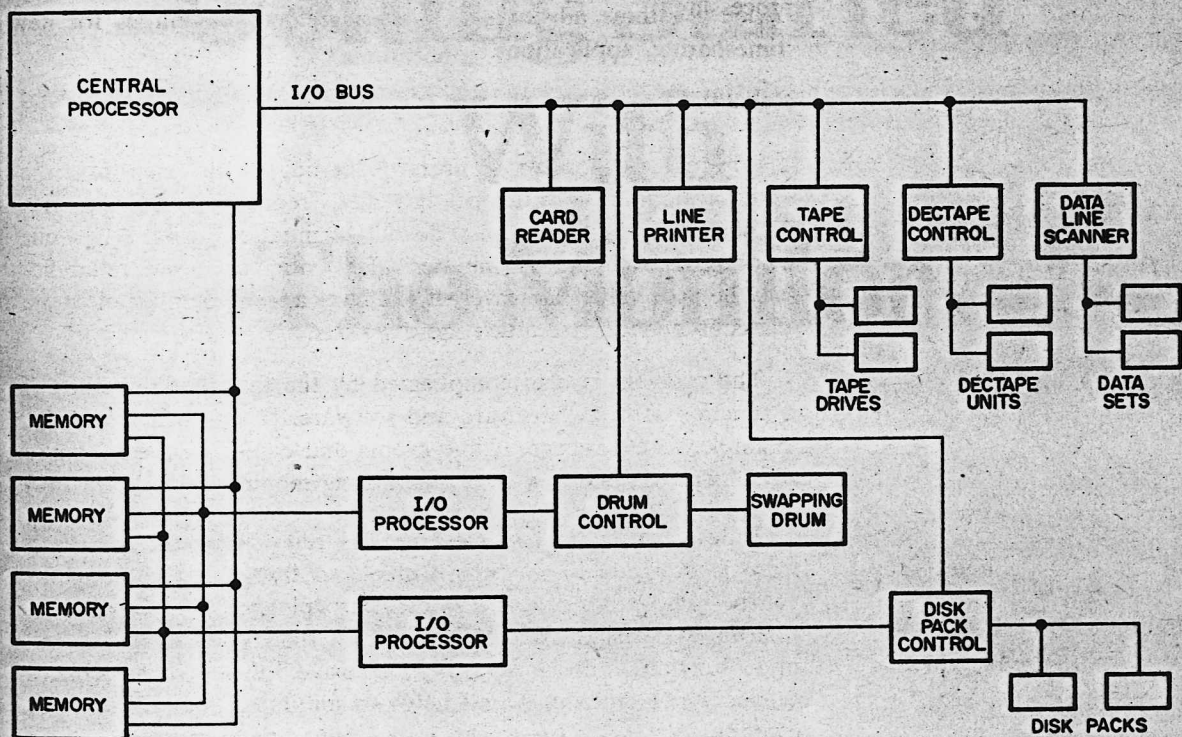
**diagnostic software**

Diagnostic software can run routinely as one of the timesharing users. Software can also maintain a log of failures, so that patterns can be established and problems remedied before serious damage occurs. Systems that employ these reliability techniques keep downtime at a minimum.



## Future of Timesharing

The advanced technology described in these pages is demonstrated by the PDP-10 systems serving timesharing users throughout the world. Typically, one of these large scale systems includes the equipment shown in the accompanying diagram—one or more swapping drums, disk packs for fast storage, magnetic tapes and DECTapes for additional secondary storage. Other peripherals include a line printer, card reader, and plotter. The data line scanner services the desired number of data sets or modems. This equipment, together with the concepts of multiprogramming, reentrant software, and advanced scheduling algorithms, provide excellent service for today's user. But tomorrow's user can expect even more.



TYPICAL PDP-10 TIMESHARING SYSTEM

### **intelligent terminal**

In a new "intelligent" terminal concept, the conventional terminal is replaced by a small computer and peripherals. The small computer will provide local computing capability and, in addition, will have direct access to the central timesharing computer when more power is required. The local system will offer line printers, card readers, and other peripherals as options.

Central processors now under development will be larger, faster, and more powerful, with the ability to serve more users at lower timesharing rates. Hardware will be more sophisticated, implementing more of the monitor's functions.

System reliability and load handling capacity will be improved through greater use of multiprocessor configurations. These configurations allow two or more central processors access to the same memory, mass storage, and peripherals.

As the user will witness, tomorrow's systems will provide better facilities, more power, faster processing, and higher reliability. And with these advances . . . even greater possibilities for new timesharing applications.



## **Book 2**

# **Getting Started With The Monitor**

Book 2

Continued

With

The Monitor

WAS FIRST PUBLISHED IN 1851 BY THE MONITOR PRESS, NEW YORK



## GETTING STARTED WITH THE MONITOR<sup>1</sup>

### 2.1 INTRODUCTION

There are basically four phases of programming: (1) writing the program, (2) inputting the program, (3) translating and loading the program, and (4) testing and debugging the program. Since the computer must be instructed in order to know what to do, the first phase is writing the program and supplying data for that program. The program may be written in a programming language that the computer is preconditioned to understand, such as BASIC, AID, COBOL, FORTRAN. A program written in the symbolic notation of one of these languages is called the source program. In the second phase of programming, the source program is inputted into the computer and stored on the disk. Although there are several ways of inputting the source program into the computer (e.g., tapes, cards), the Teletype as the device used for input and output is the main concern of this section. (See Book 7, Advanced Monitor Commands, for a discussion of other input and output devices. In the third phase, the source program is translated by the computer into a binary machine language program, and this binary program is loaded into core memory to form the core image of the translated source program. Ideally, a program should run correctly the first time, but in reality, this is not the case. A program may contain errors of many types, ranging from simple errors in typing to complex errors in the logical design of the program. Therefore, the fourth phase of programming is program testing and debugging. When errors are found, corrections are made to the source program still on the disk. The sequence of program testing and debugging is repeated until the program runs properly.

Programs are typed directly into the computer by means of the Teletype, a typewriter-like console. By typing in programs, you establish communication with other programs already resident in the computer. The first resident program you communicate with is the time-sharing

<sup>1</sup>We wish to express appreciation to Stanford University for the use of their Stanford A-1 Project User's Manual, Chapter I, SAILON No. 54, as a guide in writing the material in this section.

monitor, the most important program in the computer. (The terms monitor and system are used interchangeably to mean the time-sharing monitor.) The monitor is the master program that plays an important role in the efficient operation of the computer. Just as the Teletype is your link with the computer, the monitor is your link with the programs within the computer.

The monitor has many functions to perform, like keeping a record of what each user is doing and deciding what user should be serviced next and for how long. The one function of the monitor that is of greatest concern at this point is that the monitor retrieves any resident programs that you need. This retrieval happens only if the monitor "understands" what is expected of it. The commands to the monitor which are explained in this chapter are sufficient for the Teletype to be the device by which information is inputted into the system and by which the system outputs its results.

See section 2.10 for a discussion on How to Live With the Teletype.

## 2.2 GETTING ON THE SYSTEM

In order to gain access to the time-sharing system, you must say hello to the system by "logging in". The first move is to make contact with the computer facility by whatever means the facility has established (e.g., acoustic coupler, telephone, or dataphone). Next, notice the plastic knob (the power switch) on the lower right-hand side of the Teletype. This knob has three positions: on, off, and local (turning clockwise). When the knob is in the local position, the Teletype is like a typewriter; it is not communicating with the system at all. The knob must be turned to the on position in order to establish communication with the computer. When the Teletype is turned on, type a  $\uparrow$ C (depress the CTRL key and type C). This action establishes communication with the monitor. The monitor signifies its readiness to accept commands by responding with a period (.). All the commands discussed in this chapter can only be typed to the monitor. They are operative when the monitor has typed a period, signifying that it is waiting for a command.

The first program the monitor should call in for you is the log-in program. This is accomplished by typing LOGIN followed by a carriage-return (depress the RETURN key). All commands to the monitor are terminated with a carriage-return. When the monitor "sees" a carriage-return, it knows that a command has been typed and it begins to execute the command.

In the text, underlining is used to designate Teletype output. A carriage-return is designated by a  $\rightarrow$

By typing LOGIN, you cause the monitor to read the login program from the disk into core



memory and it is this program that is now in control of your Teletype. Before the login program is called in, the monitor assigns you a job number for system bookkeeping purposes. The system responds with an information message similar to the following.

```
JOB 17 4SP74G  
#  
_
```

In the first line, the system has assigned your job number (17) and has given the name of the monitor and its version number. This version number changes whenever a change, or patch, is incorporated into the monitor. In the second line, the number sign (#), which is typed out by the login program, signifies that it wants your identification.

The standard identification code is in the form of project numbers and programmer numbers, but individual installations may have different codes. The numbers, or whatever code each installation uses, are assigned to each user by the installation. The login program waits for you to type in your project number and your programmer number, separated by a comma and terminated with a carriage-return, following the number sign.

```
JOB 17 4SP74G  
#27,400 )  
_
```

The login program needs one more item to complete its analysis of your identification. This it requests in the next line by asking for your password.

```
JOB 17 4SP74G  
#27,400 )  
PASSWORD: )
```

Type in your password, which is also assigned by the installation, followed by a carriage-return. To maintain password security, the login program does not print the password.

If the identification typed in matches the identification stored in the accounting file in the monitor, the login program signifies its acceptance by responding with the time, date, your Teletype number, the message of the day (if any), and a period.

```
JOB 17 4SP74G  
#27,400 )  
PASSWORD: )  
1050 4-MAY-70 TTY9  
COBOL IS NOW AVAILABLE ON THE SYSTEM  
.
```

This typeout indicates that the login program has exited and returned control to the monitor. You have successfully logged in and may now have the monitor call in other programs for you. If the identification typed in does not match the identification in the accounting file, the monitor types out the error message

### ?INVALID ENTRY-TRY AGAIN

#  
—

If this error message occurs, type in the correct project-programmer numbers and password.

## 2.3 FILES

When you want to run a program, first type in the program and decide on a name for it. The program is stored on the disk with the specified name. Then translate the program by calling in a translator and giving it the name of the program you wish to translate.

A program, or data, is stored on the disk in files. If a program is being typed in to a text editor (for example, TECO), the editor is busy accepting the characters being typed in and generating a disk file for them. Then, when the program is to be translated, the translator reads this file just created and generates a relocatable binary file. Since you may have many files and the other users on the computer may have files, there must be a method for keeping all of these files separate. This is accomplished by giving each user a unique area on the disk. This area is identified by your project and programmer numbers. For example, if your project and programmer numbers are 27,400, you have a disk area by that name. Each file you create goes to your disk area and must be uniquely named.

Files are named with a certain convention, the same as a person is named. The first name, the filename, is the actual name of the file, and the last name, the filename extension, indicates what group the file is associated with. The filename and the filename extension are separated by a period.

Filenames are from one to six letters or digits. All letters or digits after the sixth are ignored. The filename extension is from one to three letters or digits. It is generally used to indicate file format. The following are examples of standard filename extensions.



|      |                                    |
|------|------------------------------------|
| .TMP | Temporary file                     |
| .MAC | Source file in MACRO language      |
| .F4  | Source file in FORTRAN IV language |
| .BAS | Source file in BASIC language      |
| .CBL | Source file in COBOL language      |
| .REL | Relocatable binary file            |
| .SAV | A saved core image                 |

Since files are identified by the complete name and the project and programmer numbers, two users may use the same filename as long as they have different project and programmer numbers; the files would be distinct and separate. The following are examples of filenames with file-name extensions.

|            |                                      |
|------------|--------------------------------------|
| MAIN.F4    | A FORTRAN file named MAIN            |
| SAMPLE.BAS | A BASIC file named SAMPLE            |
| TEST1.TMP  | A temporary file named TEST 1        |
| NAME.REL   | A relocatable binary file named NAME |

## 2.4 CREATING FILES<sup>1</sup>

The two commands mentioned in this section use two editors to create a new disk file. One of the editors is LINED, a disk-oriented editor, and the other is TECO, the Text Editor and Corrector (see Book 8 for discussion of both editors). Each command requires a filename as its argument and should have a filename extension. A new file may be created with either of these commands, depending on the editor desired. If line numbers are desired, LINED should be used, since TECO generates a non-sequence numbered file.

### 2.4.1. The CREATE Command

The CREATE command is used only to create a new disk file. When this command is executed, the monitor calls in LINED to initialize a disk file with the specified name and to accept input from the Teletype. At this point, begin to type in your program, line by line. LINED types a line number at the beginning of each line so that later a reference to a given line may be made in order to make corrections. Below is a sample program using the commands discussed so far.

<sup>1</sup> A BASIC or AID user does not need the following sections. These two compilers have built-in facilities to create and edit files. See Book 3 for BASIC and Book 4 for AID.

↑C

Establish communication with the monitor. Type C while depressing the CTRL key.

LOGIN )

Begin the login procedure.

JOB 17 4SP74G

Job number assigned, followed by monitor name and version. Login program requests identification (project number and programmer number).

#  
—

27,400 )

Type in project-programmer number.

PASSWORD: )

Login program requests password. Type it in; it is not printed.

1050 4-MAY-70 TTY9  
COBOL IS NOW AVAILABLE  
ON THE SYSTEM

If identification matches identification stored in the system, the monitor responds with the time, date, Teletype number, message of the day, and a period.

⋮

CREATE MAIN.F4 )

A new file on the disk is to be created and called MAIN.F4. The extension .F4 is used because the program is to be a FORTRAN source file. LINED is called in to create the file.

\*  
—

Response from LINED signifying it is ready to accept commands.

I )

A command to LINED to insert line numbers starting with 10 and incrementing by 10 (see Book 8).

00010 TYPE 53 )

Type in your FORTRAN PROGRAM.

00020 53 FORMAT (' THIS IS MY PROGRAM' ) )

00030 END )

00040 \$

The Ⓢ (altmode) is a command to LINED to end the insert. On the Teletype this key is labeled ALT, ESC, or PREFIX.

\*  
—

Response from LINED signifying it is ready to accept another command.

E )

A command to LINED to end the creation of the file.

\*  
—

Response from LINED indicating readiness to accept a command.

↑C

Return to the Monitor.

⋮

The monitor now has control of the program



The three LINED commands (I, altmode, E) shown in the example are fully discussed in Book 8.

## 2.4.2 The MAKE Command

This command can also be used to open a new disk file for creation. It differs from the CREATE command in that TECO is used instead of LINED. (TECO is discussed in Book 8.) Otherwise, the CREATE and MAKE commands operate in the same manner.

```
.MAKE FILEA.F4 )  
* I (Text input)$$  
EX$$  
EXIT  
↑C  
:
```

The altmode (\$) and the EX command are commands to TECO and are explained in the TECO section of Book 8.

## 2.5 EDITING FILES

After creating a text file, you may wish to modify, or edit, it. The following two commands cause an existing file to be opened for changes. One command (EDIT) calls in LINED, and the other (TECO) calls in TECO. In general, the editor used to create the file should be used for editing. Each command requires, as its argument, the same filename and filename extension used to create the file.

### 2.5.1 The EDIT Command

The EDIT command causes LINED to be called in and, as the name implies, signifies that you wish to edit the specified file. LINED responds with an asterisk and waits for input. The file specified must be an already existing sequence-numbered file on the disk. For example, in section 2.4.1, the file MAIN.F4 was created. If the command

```
. EDIT MAIM.F4 )
```

is given to edit the file, the computer responds with an error message (assuming that there was no file named MAIM.F4). The command

```
.EDIT MAIN.F4 )
```

causes the right file to be opened for editing.

### 2.5.2 The TECO Command

The TECO command is similar to the EDIT command except that it causes the TECO program to open an already existing non-sequence-numbered file on the disk for editing purposes. The command sequence

```
._ TECO FILEA.F4 )  
._ * (editing)$$  
._ * EXIT$$  
._
```

causes TECO to open FILEA.F4 for editing and close the file upon completion, creating a backup file out of the original file. Whenever one of the commands used to create or edit a file is executed, this command with its arguments (filename and filename extension) is "remembered" as a temporary file on the disk. Because of this, the file last edited may be recalled for the next edit without having the filename specified again. For example, if the command

```
._ CREATE PROG1.MAC )
```

is executed, then you may type the command

```
._ EDIT )
```

instead of

```
._ EDIT PROG1.MAC )
```

assuming that no other CREATE, TECO, MAKE, or EDIT command was used in-between. As mentioned before, if a command tries to edit a file that has not been created, an error message is given.

## 2.6 MANIPULATING FILES

You may have many files saved on your disk area. (For discussion on how to save a file on your disk area, see Book 7.) The list of your files, along with lists of other users' files, is



kept on the disk in what are called user directories. Suppose you cannot remember if you have created and saved a particular file. The next command helps in just that type of situation.

### 2.6.1 The DIRECTORY Command

The DIRECTORY command requests from the monitor a listing of the directory of your disk area. The monitor responds by typing on the Teletype the names of your files, the date on which each file was created, and the length of each file in PDP-10 disk blocks. A disk block consists of 128<sub>10</sub> PDP-10 words. Names of files not explicitly created by you may show up in the directory. These files were created as intermediate files for storage by programs you may have used. For example, in translating a file, the translator generates a file with the same filename but with a filename extension of .REL. This file contains the relocatable binary translation of the source file. You may also notice filenames with the filename extension of .TMP. This extension signifies a temporary file created and used by different CUSPs.

### 2.6.2 The TYPE Command

By listing your directory on the Teletype, you know the names of the files on your disk area. But what if you have forgotten the information contained in a particular file? The TYPE command causes the contents of source files specified in your command string to be typed on your Teletype. For example, the command

```
_TYPE MAIN.F4 )
```

causes the file MAIN.F4 to be typed on the Teletype. Multiple files separated by commas may be specified in one command string, and only source files, not binary files, may be listed.

This command allows the "asterisk construction" to be used. This means that the filename or the filename extension may be replaced with an asterisk to mean any filename or filename extension. For example, the command

```
_TYPE FILEB.* )
```

causes all files named FILEB, regardless of filename extensions, to be typed. The command

```
_TYPE *.MAC )
```

causes all files with the filename extension of .MAC to be typed. The command

`_TYPE *.* )`

causes all files to be typed.

### 2.6.3 The DELETE Command

Having finished with a file, you may erase it from your disk area with the DELETE command. Multiple files may be deleted in one command string by separating the files with commas. For example,

`_DELETE LINEAR )`

and

`_DELETE CHANGE.F4, SINE.REL )`

are both legal commands. The asterisk convention discussed in section 2.6.2 may also be used with the DELETE command.

### 2.6.4 The RENAME Command

The names of one or more files on your disk area may be changed with the RENAME command. The old filename on the right and the new filename on the left are separated by an equal (=) sign. In renaming more than one file, each pair of filenames (new=old) is separated by commas. For example, the command

`_RENAME SALES.CBL=GROSS.CBL, FILE2.F4=FILE1.F4 )`

changes the name of file GROSS.CBL to SALES.CBL and file FILE1.F4 to FILE2.F4. The old filename no longer appears in your directory; instead the new filenames appear containing exactly the same data as in the old files. The asterisk convention may again be used. For example, the command

`_RENAME *.F4= * )`

causes all files with no filename extension to have the extension .F4.

## 2.7 TRANSLATING, LOADING, EXECUTING, DEBUGGING PROGRAMS

At this point you know how to get on the system, how to create and edit a source file of a



program, and how to list your source file on the Teletype. The program has not been executed. This only happens after it has been translated into the binary machine language understandable to the computer and loaded into core memory. More often than not the program must be debugged.

### 2.7.1 The COMPILE Command

This command has as its argument one or more filenames separated by commas. It causes each command to be processed (translated) if necessary by the appropriate processor (translator). It is considered necessary to process a file if no .REL file of the source file exists, or if the .REL file was created before the last time the source file was edited. If the .REL file is up-to-date, no translation is done. The appropriate processor is determined by examining the extension of the file. The following shows which processor is used for various extensions.

|                              |                       |
|------------------------------|-----------------------|
| . MAC                        | MACRO assembler       |
| . F4                         | FORTAN IV compiler    |
| . CBL                        | COBOL compiler        |
| . REL                        | No processing is done |
| other than above,<br>or null | "Standard processor"  |

The standard processor is used to translate programs with null or nonstandard extensions. The standard processor is FORTRAN at the beginning of the command string, but may be changed by use of various switches (See the PDP-10 Reference Handbook, Communication With the Monitor). Although it is not necessary to indicate the extension of a file in the COMPILE command string, the standard processor can be disregarded if all source files are kept with the appropriate extension.

When the appropriate translator has translated the source file, there is a file on your disk area with the filename extension .REL and the same filename as the source file. This file is where the translator stores the results of its translation and is called the relocatable binary of the program. The program is now translated into binary machine language, but is still on the disk. Since the disk is used for storage and not for execution, a copy of the binary program must be loaded into core memory to form a core image. The core memory of the computer is used for execution; it is like a scratch pad. The COMPILE command does not generate a core image, but the following three commands do.

### 2.7.2 The LOAD Command

The LOAD command performs the same operations as the COMPILE command and in addition causes the Linking Loader to be run. The Linking Loader is a resident program that takes the specified REL files, links them together, and generates a core image. The LOAD command does not cause execution of the program.

### 2.7.3 The EXECUTE Command

This command performs the functions of the LOAD command and also begins execution of the loaded programs, if no translation or loading errors are detected. The compiled program is now in core memory and running, and what happens next depends on the program. More than likely, the program is not returning the correct answers, and you now enter the magic world of program debugging.

### 2.7.4 The DEBUG Command

This command prepares for the debugging of a program in addition to performing the functions of the COMPILE and LOAD commands. DDT, the Dynamic Debugging Technique program (see the DDT section in the PDP-10 Reference Handbook), is loaded into core memory first, followed by the program. Upon completion of loading, DDT is started rather than the program. A command to DDT may then be issued to begin the program execution. This command should be used by the experienced programmer familiar with DDT. The above four commands have extended command forms discussed in the PDP-10 Reference Handbook.

The following is an example showing the compilation and execution of a FORTRAN main program and subroutine. The login procedure is not shown.

|                                                 |                                                                    |
|-------------------------------------------------|--------------------------------------------------------------------|
| <u>._</u> CREATE MAIN.F4 )                      | CREATE a disk file                                                 |
| <u>*I</u> )                                     | Command to LINED to begin inserting on line 10, incrementing by 10 |
| 00010 TYPE 69 )                                 | Statements of the FORTRAN main program                             |
| 00020 69 FORMAT (' THIS IS THE MAIN PROGRAM') ) |                                                                    |
| 00030 CALL SUB1 )                               |                                                                    |
| 00040 END )                                     |                                                                    |
| 00050 \$                                        | Altmode ends the insert                                            |
| <u>*E</u> )                                     | LINED command to end the edit                                      |
| <u>*↑C</u>                                      | Return to the Monitor                                              |
| <u>._</u> CREATE SUB1.F4 )                      | Create a disk file for the subroutine                              |
| <u>*I</u> )                                     | Begin inserting at line 10 incrementing by 10                      |



00010 SUBROUTINE SUBR ) Statements of the FORTRAN Subroutine  
 00020 TYPE 105 )  
 00030 105 FORMAT (' THIS IS SUB1') )  
 00040 RETURN )  
 00050 \$ Altmode ends the insert  
 \* E ) LINED command to end the edit.  
 \* ↑ C Return to Monitor  
 . EXECUTE MAIN.F4, SUB1.F4 ) Request execution of the programs created  
 FORTRAN: MAIN.F4 FORTRAN reports its progress  
 FORTRAN: SUB1.F4

# LOADING

## 000001 UNDEFINED GLOBALS

SUB1 000152 There is no subroutine named SUB1

?

LOADER 3K CORE This includes the space for the loader.

? EXECUTION DELETED No execution was done

EXIT

↑ C

. EDIT ) Ask to edit SUB1.F4, filename need not be mentioned since it was the last file edited.

\* P10,20 ) Type lines 00010 and 00020 on the Teletype.

00010 SUBROUTINE SUBR

00020 TYPE 105

\* I10 ) Insert a new line 10

00010' SUBROUTINE SUB1 )

\* E )

\* ↑ C

. EXECUTE MAIN.F4, SUB1.F4 ) Request execution

FORTRAN: SUB1.F4 Only the subroutine is recompiled since only it has been edited.

# LOADING

Both MAIN and SUB1 are loaded

LOADER 3K CORE

# EXECUTION

THIS IS THE MAIN PROGRAM Execution begins

THIS IS SUB1

EXIT

↑ C

Execution ends

.

463-1156

## 2.8 GETTING INFORMATION FROM THE SYSTEM

There are several monitor commands that are used to obtain information from the system. Three commands useful at this point are discussed in this section, and additional commands are discussed in Book 7, Advanced Monitor Commands.

### 2.8.1 The PJOB Command

If you have forgotten the job number assigned to you at login time, you may use the PJOB command to obtain it. The system responds to this command by typing out your assigned job number. For example,

```
_ PJOB )  
17
```

### 2.8.2 The DAYTIME Command

This command gives the date followed by the time of day. The time is presented in the following format:

hh:mm

where hh represents the hours and mm represents the minutes. For example,

```
_ DAYTIME )  
17-JUNE-70 14:37
```

### 2.8.3 The TIME Command

The TIME command produces three lines of typeout. The first line is the total running time since the last TIME command was typed. The second line is the total running time since you logged in. The third line is used for accounting purposes. The time is presented in the following format:

hh:mm:ss.hh

where hh represents the hours, mm the minutes, and ss.hh the seconds to the nearest hundredth. For example,



TIME )

52.45

02:29.95

KILO-CORE-SEC=57

In the first two lines, you are told that you have been running 52.45 seconds since the last time you typed the TIME command, and a total of 2 minutes and 29.95 since you logged in. The third line of typeout is used by your installation for accounting and is the integrated product of running time and core size. See the PDP-10 Reference Handbook, Communicating With the Monitor.

## 2.9 LEAVING THE SYSTEM

Now that you know how to log into the system and create and run a program, you might be wondering how you leave the system. You have to tell the system you are leaving, and you do this by the KJOB command.

### 2.9.1 The KJOB Command

The KJOB command is your way of saying goodbye to the system. Many things happen when you type the command. The job number assigned to you is released and your Teletype is now free for another user. An automatic TIME command is performed. In addition, if you have any files on your disk area, the monitor responds with

CONFIRM:

and you have several options available to you. By typing a carriage-return after the CONFIRM: message, the monitor lists the options available. For example, the following typeout occurs by responding to the confirm message with a carriage return.

TYPE ↑C TO ABORT LOG-OUT; OR

TYPE ONE OF THE FOLLOWING ( AND CAR RET):

K TO KILL JOB AND DELETE ALL UNPROTECTED FILES;

L TO LIST YOUR DISK DIRECTORY; OR

I TO INDIVIDUALLY SAVE AND DELETE FILES AS FOLLOWS:

AFTER EACH FILE NAME IS LISTED, TYPE:

P TO SAVE AND PROTECT,

S TO SAVE WITHOUT PROTECTING, OR  
CAR RETURN ONLY TO DELETE.

CONFIRM:

You may now use the options available. If K was used as the option, the following is a sample of what is output ted to your Teletype.

JOB 33, USER [27,560] LOGGED OFF TTY34 1317 20-FEB-70  
DELETED ALL 2 FILES (INCLUDING UFD, 3. DISK BLOCKS)  
RUNTIME 0 MIN, 00.29 SEC

Remember that the CONFIRM message is typed only if there are files on your disk area. If there are no files on your disk area, the typeout would look like the following:

\_ KJOB )  
JOB 17, USER [27,3201] LOGGED OFF TTY17 1317 20-FEB-70  
RUNTIME 0 MIN, 00.29 SEC

## 2.10 HOW TO LIVE WITH THE TELETYPE

On the Teletype, there is a special key marked CTRL called the Control Key. If this key is held down and a character key is depressed, the Teletype types what is known as a control character rather than the character printed on the key. In this way, more characters can be used than there are keys on the keyboard. Most of the control characters do not print on the Teletype, but cause special functions to occur, as described in the following sections.

There are several other special keys that are recognized by the system. The system constantly monitors the typed characters and, most of the time, sends the characters to the program being executed. The important characters not passed to the program are also explained in the following sections. (See also the PDP-10 Reference Handbook, Communicating With the Monitor.)

### 2.10.1 Control - C

Control - C ( $\uparrow$ C) interrupts the program that is currently running and takes you back to the monitor. The monitor responds to a control - C by typing a period on your Teletype, and you



may then type another monitor command. For example, suppose you are running a program in BASIC, and you now decide you want to leave BASIC and run a program in AID. When BASIC requests input from your Teletype by typing an asterisk, type control - C to terminate BASIC and return to the monitor. You may now issue a command to the monitor to initialize AID (.R AID). If the program is not requesting input from your Teletype (i.e., the program is in the middle of execution) when you type control - C, the program is not stopped immediately. In this case, type control - C twice in a row to stop the execution of the program and return control to the monitor. If you wish to continue at the same place that the program was interrupted, type the monitor command CONTINUE. As an example, suppose you want the computer to add a million numbers and to print the square root of the sum. Since you are charged by the amount of processing time your program uses, you want to make sure your program does not take an unreasonable amount of processing time to run. Therefore, after the computer has begun execution of your program, type control - C twice to interrupt your program. You are now communicating with the monitor and may issue the monitor command TIME to find out how long your program has been running. If you wish to continue your program, type CONTINUE and the computer begins where it was interrupted.

#### 2.10.2 The RETURN Key

This key causes two operations to be performed: (1) a carriage-return and (2) an automatic line-feed. This means that the typing element returns to the beginning of the line (carriage-return) and that the paper is advanced one line (line-feed). Commands to the monitor are terminated by depressing this key.

#### 2.10.3 The RUBOUT Key

The RUBOUT key permits correction of typing errors. Depressing this key once causes the last character typed to be deleted. Depressing the key n times causes the last n characters typed to be deleted. RUBOUT does not delete characters beyond the previous carriage-return, line-feed, or altmode. Nor does RUBOUT function if the program has already processed the characters you wish to delete.

The monitor types the deleted characters, delimited by backslashes. For example, if you were typing CREATE and go as far as CRAT, you can correct the error by typing two RUBOUTS and then the correct letters. The typeout would be

CRAT\TA\EATE

Notice that you typed only two RUBOUTS, but \TA\ was printed. This shows the deleted characters, but in reverse order.

#### 2.10.4 Control - U

Control - U ( $\uparrow$ U) is used if you have completely mistyped the current line and wish to start over again. Once you type a carriage-return, the command is read by the computer, and line-editing features can no longer be used on that line. Control - U causes the deletion of the entire line, back to the last carriage-return, line-feed, or altmode. The system responds with a carriage-return, line-feed so you may start again.

#### 2.10.5 The ALTMODE Key

The ALTMODE key, which is labeled ALTMODE, ESC, or PREFIX, is used as a command terminator for several programs, including TECO and LINED. Since the ALTMODE is a non-printing character, the Teletype prints an ALTMODE as a dollar sign (\$).

#### 2.10.6 Control - O

Control - O ( $\uparrow$ O) tells the computer to suppress Teletype output. For example, if you issue a command to type out a 100 lines of text and then decide that you do not want the typeout, type control - O to stop the output. Another command may then be typed as if the typeout had terminated normally.

#### 2.10.7 Control - B

Control - B ( $\uparrow$ B) affects the printing of Teletype output in one of two ways depending on your Teletype; it either restores printing of the characters or suppresses double printing of the characters. Suppose that when you begin typing on the Teletype, you notice that the characters you are typing are not printing on the Teletype paper. Type control - B to restore the printing of the characters. On the other hand, suppose you receive double printing of your typeins. To suppress this double printing, type control - B.

#### 2.10.8 Control - F

This control character is needed only for the KSR37 Teletype. This key changes the way lower case characters are handled by the system. Normally, the system converts all lower case letters to upper case. Since the KSR37 Teletype is capable of transmitting both lower and upper case letters, control - F is used to permit the entry of lower case letters.



## Book 3

# Conversational Programming With BASIC





# **Book 7**

## **Advanced Monitor Commands**



ADVANCED

COMMAND

Book 7

Monitor

Commands

The logical device name is an alias for the physical device name. The logical device name is an alias for the physical device name.



## ADVANCED MONITOR COMMANDS

The system controls many peripheral devices, such as Teletypes, magnetic tape drives, DECtape drives, card readers and punches, line printers, papertape readers and punches, and disks. The monitor is responsible both for allocating these peripheral devices, as well as other system resources (e.g., core memory), and for maintaining a pool of such available resources from which you can draw.

Each device controlled by the system has a physical name associated with it. The physical name is unique. It consists of three letters and zero to three numerals specifying a unit number. The following table lists the physical names associated with various peripheral devices.

Table 7-1  
Peripheral Devices

| Device            | Physical Name             |
|-------------------|---------------------------|
| Teletype          | TTY0, TTY1, . . . , TTY77 |
| Console Teletype  | CTY                       |
| Paper Tape Reader | PTR                       |
| Paper Tape Punch  | PTP                       |
| Plotter           | PLT                       |
| Line Printer      | LPT                       |
| Card Reader       | CDR                       |
| Card Punch        | CDP                       |
| DECtape           | DTA0, DTA1, . . . , DTA7  |
| Magnetic Tape     | MTA0, MTA1, . . . , MTA7  |
| Disk              | DSK                       |

You may also give each device a logical device name. The logical device name is an alias, and the device can be referred to either by this alias or by the physical name. The logical

name consists of one to six alphanumeric characters of your choice. The reason for logical device names is that in writing a program you may use arbitrarily selected device names (logical device names) that can be assigned to the most convenient physical devices at runtime. However, care should be exercised in assigning logical device names because these names have priority over physical device names. For example, if a DECTape is assigned the logical name DSK, then all of your programs attempting to use the disk via the physical name DSK end up using the DECTape instead. It is wise not to give any device the logical name DSK because certain monitor commands (such as the COMPILE commands) make extensive use of special features that the disk has but other devices do not have. The following examples show the use of logical and physical device names.

`._ ASSIGN DTA,ABC )`

Assign a DECTape the logical name ABC.

`._ ASSIGN MTA1,XYZ )`

Assign magnetic tape drive #1 the logical name XYZ.

`._ ASSIGN PTR,FOO )`

Assign the papertape reader the logical name FOO.

In order to use most peripheral devices, you must assign the desired device to your job. You may assign a device either by a program or from the console. The first kind of assignment occurs when you write a program that uses a particular device. When the program begins using the device, it is assigned to you on a temporary basis and released from you when your program has finished with it. The second kind of assignment occurs when you explicitly assign the device by means of the ASSIGN monitor command. This is a permanent assignment that is in effect until the device is released by a DEASSIGN or FINISH monitor command or by your logging off the system.

When you assign a device to your job, the monitor associates your job number with that device. This means that no other user may use the device while you are using it. The only exception is the disk, which is accessible by all users. When you assign the disk, you are allocated a fraction of the disk, not the entire unit. When you deassign a device or kill your job, the device is returned to the monitor's pool of available resources.

## 7.1 COMMANDS TO ALLOCATE SYSTEM RESOURCES

### 7.1.1 The ASSIGN Command

The ASSIGN command is used to assign a peripheral device on a permanent basis for the duration of your job or until you explicitly deassign it. This command must have as an argument the legal physical device name (see Table 7-1) of the device you wish to assign. For example, if



you want to assign a DECtape drive to your job, type

**.\_ ASSIGN DTA )**

The monitor responds with the message

**DTAn ASSIGNED**

where n is the unit number of the DECtape drive assigned to your job. If all drives are in use, the monitor responds with

**NO SUCH DEVICE**

and you must wait until a drive becomes available. You may assign a specific DECtape drive as follows:

**.\_ ASSIGN DTA3 )**

The monitor responds with

**DTA3 ASSIGNED**

if the drive is available, or

**ALREADY ASSIGNED TO JOB n**

if job n is using DECtape drive #3.

The ASSIGN command may also have, as an optional argument, a logical device name following the physical device name. The logical device name may be used in place of the physical device name in all references to the device. For example, if you want to use DECtape drive #1 and have it named SAMPLE, type the command

**.\_ ASSIGN DTA1 SAMPLE )**

If DECtape drive #1 is free, the monitor responds with

**DTA1 ASSIGNED**

and stores the logical name you typed. You may then refer to the DECtape by the name SAMPLE until you explicitly release the device or kill your job.

Logical names can be very useful. Suppose you write a program that uses DECtape drive #5 and refers to it by its physical name (DTA5). When you run your program, you find that DECtape drive #3 is the only drive available. Instead of rewriting your program to use DECtape drive #3, type

.\_ ASSIGN DTA3 DTA5 )

Thereafter, whenever your program refers to DTA5, it is actually referring to DTA3. Since logical device names are strictly your own, they are different from the logical names of other users. The following is an example using physical and logical device names.

.\_ ASSIGN DTA,NAME )

Assign a DECtape drive the logical name NAME.

DEVICE DTA4 ASSIGNED

DECtape drive #4 has been assigned.

.\_ ASSIGN DTA,LINE )

Find another DECtape drive; assign the logical name LINE.

NO SUCH DEVICE

All DECtape drives are in use.

.\_ ASSIGN PTP,NAME )

Reserve paper tape punch.

LOGICAL NAME ALREADY IN USE

DEVICE PTP ASSIGNED

Paper tape punch is assigned but NAME still refers to DTA4 only.

.\_ ASSIGN DTA3,LINE )

Request DECtape drive #3 and give it the logical name LINE

ALREADY ASSIGNED TO JOB7 Another user (job 7) has DTA3.

### 7.1.2 The DEASSIGN Command

The DEASSIGN command is used to release one or more devices currently associated with your job. This command may have as an argument a physical or logical device name. If an argument is given, the specified devices are released. If an argument is not specified, all devices assigned to your job are released. When devices are released, they are returned to the monitor's pool of available resources for use by other users. The DEASSIGN command does not affect any temporary assignments your job may have for devices.



### 7.1.3 The REASSIGN Command

The REASSIGN command allows you to give a device assigned to you to another user without having the device returned to the monitor's pool of available resources. Two arguments are required with this command: the name of the device being reassigned and the job number of the user who is receiving the device. For example, suppose you have finished with DECTape drive #6 and the person who is job 10 wants it. Type the command

```
_ REASSIGN DTA6 10 )
```

This deassigns DECTape drive #6 from your job and assigns it to job 10, just as if you had typed

```
_ DEASSIGN DTA6 )
```

and job 10 had typed

```
_ ASSIGN DTA6 )
```

immediately thereafter. All devices except a Teletype console can be reassigned.

### 7.1.4 The FINISH Command

The FINISH command is used to prematurely terminate a program that is being executed while preserving as much output as possible. If this command is not used, part or all of the output file may be lost. The FINISH command may be followed by a physical or logical device name, in which case any input or output currently in progress in relation to that device is terminated. If no device is specified, input or output is terminated on all devices assigned to your job. The monitor responds to this command by terminating output, closing the file, and releasing the device for use by others.

This command could be used if you were generating an assembly listing of a program on your disk area and decided that you wanted only the first part of the listing, not the entire listing. Type

```
↑ C  
_ FINISH DSK )
```

and the monitor completes the writing of your listing and releases the disk.

### 7.1.5 The CORE Command

The CORE command allows you to modify the amount of core assigned to your job. The command is followed by a decimal number representing the total number of 1K blocks (1024 word blocks) that you want the program to have from this point on. For example, if you want the program to have 8K blocks of core, type

```
. CORE 8 )
```

and the monitor gives the program 8K blocks, if available. If you request additional core and there is none available, the monitor responds with an error message. If the CORE command is followed by the decimal number 0, your program disappears from core because you are requesting 0K blocks of core. If the decimal number following the command is omitted, the monitor types out ( 1 ) the total number of 1K blocks you have, ( 2 ) the maximum you can request, and ( 3 ) the amount of core not assigned to any user.

## 7.2 COMMANDS TO MANIPULATE TELETYPE ASSIGNMENTS

### 7.2.1 The TALK Command

The TALK command allows you to type directly to another user's console. This means that everything you type appears on his Teletype and everything he types appears on yours. If his console is not both talking to the monitor and positioned at the left margin, you get a BUSY message. This command is especially useful when you are at a remote Teletype and wish to direct people at the computer site to mount a DECtape for you. When you finish talking to the other console, type control - C to get back to the monitor.

### 7.2.2 The DETACH Command

The DETACH command causes your Teletype to be disconnected from your program and released to control another job. This means that, while your program is disconnected, you may log in again, receive a new job number, and do something else. The job that was detached from your Teletype is said to be a background job. This means that it is not under control of any user's console. If your background job attempts to type something to the Teletype, it is stopped, for there is no Teletype attached to it.

### 7.2.3 The ATTACH Command

The ATTACH command allows you to attach a console to a background, or detached job. You must specify the number of the job to which you wish to attach. If you are the owner of the



detached job, your console is immediately detached from your current job and attached to your detached job. After this command is executed, the console is in communication with the monitor. If the job you just attached to happens to be running, type CONTINUE without affecting the status of the job.

If you are not the owner of the detached job, you must also specify the project-programmer number of the owner. The project-programmer number must be enclosed in square brackets (e.g. [27,400]) for this command to work. If the job whose job number you typed is already attached to a Teletype, you cannot attach and the monitor responds with

#### TTYn ALREADY ATTACHED

where n is the number of the Teletype attached to the job. Observe that only one Teletype can be attached to a job at any time. (For a demonstration on attaching and detaching jobs, see the PDP-10 Reference Handbook.)

### 7.3 COMMANDS TO REQUEST LINE PRINTER OUTPUT

In Book 2, the TYPE command for listing source files on your Teletype was discussed. In addition, there are three commands that may be used to list files on the line printer: LIST, CREF, and DIRECTORY.

#### 7.3.1 The LIST Command

The LIST command causes the contents of the specified source files to be typed on the line printer. Headings at the top of each page tell the page number, the name of the file, and the date of printing. You may list files from disk or DECTape. For example, the command

```
_ LIST TEST.F4 )
```

causes your disk file TEST.F4 to be listed on the line printer. You may list multiple files by separating the filenames by commas. For example, if you want to list three files, SAMPLE.BAS, FILEX.F4, MAIN.F4, from DECTape drive #4, type the command

```
_ LIST DTA4: SAMPLE.BAS, FILEX.F4, MAIN.F4 )
```

The asterisk convention discussed in Book 2 may be used with this command. For example,

```
_ LIST *.F4 )
```

causes all files with the filename extension .F4 to be listed.

### 7.3.2 The CREF Command

The CREF command is used to list a certain type of file called a cross-reference file. This command is an invaluable aid in program debugging. If a COMPILE, LOAD, EXECUTE, or DEBUG command string (see Book 2) has a /CREF switch, the command string generates an expanded listing that includes (1) the original code as it appears in the file, (2) the octal values the code represents, (3) the relative locations into which the octal values go, (4) a list of all the symbols your program uses, and (5) the numbers of the lines on which each symbol appears. This is called a cross-reference listing. To print this listing file, you must call in a special cross-reference lister with the CREF command. All the cross-reference listing files you have generated since the last CREF command are printed on the line printer. The file containing the names of the cross-reference listing files is then deleted so that subsequent CREF commands will not list them again.

### 7.3.3 The DIRECTORY Command

When a DTAn: argument is specified with the DIRECTORY command, the directory of DECTape n is typed on the Teletype. (See Book 2 for a discussion of the DIRECTORY command when no argument is specified.) For example, the command

```
. DIRECTORY DTA2 )
```

types the directory of DECTape drive #2 on the Teletype.

Besides having optional device arguments, this command has two switch options. The first switch option is /F. Including /F in the command string causes the short form of the directory to be listed on the Teletype. The short form of the directory consists of the names of your files and the length of each file in PDP-10 disk blocks. (The long form of the directory also lists the creation dates of each file.) The second switch option is /L. Including /L in the command string causes the output of the directory to go to the line printer rather than to the Teletype. For example, the command

```
. DIRECTORY /L )
```

lists your directory of your disk area on the line printer. The line printer is assigned to you on a temporary basis and is released when the output is finished.



## 7.4 COMMANDS TO MANIPULATE CORE IMAGES

By using one of the following commands, you can load core image files (see Book 2 for the definition of a core image file) from disk, DECTapes, and magnetic tapes into core and then later save the core images. These files can be retrieved and controlled from the user's console. Files on disk and DECTape are called by filename, and if you have any files on magnetic tape, you must position the tape to the beginning of the file.

### 7.4.1 The SAVE Command

The SAVE command causes your current core image to be saved on the specified device with the specified filename. This command must be followed by several arguments. First, you must tell the monitor the device on which you want to save the core image. A colon may follow the device name. Second, you must give a name to the core image file. If the filename extension is not specified, the monitor designates one. You may specify the amount of core in which you want your file saved by specifying a decimal number to represent the number of 1K blocks. For example, if you want to save your core image on DECTape drive #2, give it the name SALES, and allow 12K of core for storage, type

```
SAVE DTA2: SALES 12 )
```

A file called SALES is created and your core image is stored in it. If you list your DECTape directory, the length of the file is slightly over 12,000 words. After you use this command, you cannot continue executing the program. The program can be restarted only from the beginning.

### 7.4.2 The RUN Command

The RUN command allows you to run programs you previously saved on the disk, DECTape, or magnetic tape. This command reads the core image file from a storage device and starts its execution. You must specify the device containing the core image file and the name of that file. The file must have been saved previously with a SAVE command. If the file is not a saved program, the monitor responds with an error message. If the core image file you want to execute is on another user's disk area, you must specify his project-programmer number, enclosed in square brackets. Again, you may specify the amount of core to be assigned to the program if different from the minimum core needed to load the program or from the core argument of the SAVE command.

### 7.4.3 The R Command

The R command is a special form of the RUN command. This command runs programs (or CUSPs) that are part of the system, rather than programs that are your own. The R command is the usual way to run a CUSP that does not have a direct monitor command associated with it. For example, the only way to run BASIC and AID is by the commands

```
._R BASIC )  
and  
._R AID )
```

A device name or a project programmer number may not be specified for this command.

### 7.4.4 The GET Command

The GET command is the same as the RUN command except that it does not start the program; it merely generates a core image and exits. The monitor types

#### JOB SETUP

and is ready to accept another command.

## 7.5 COMMANDS TO START A PROGRAM

### 7.5.1 The START Command

The START command begins execution of the program at its starting address, the location specified within the file, and is valid only if you have a core image. This command allows you to specify another starting address by typing the octal address after the command. Normally, to start a program, type

```
._ START )
```

but to start a program at the specified octal location 347, type

```
._ START 347 )
```

A GET command followed by a START command is equivalent to a RUN command.



### 7.5.2 The HALT ( ↑C) Command

Typing ↑C stops your program and takes you back to the monitor. The program "remembers" at what point it was interrupted so that it may subsequently be continued. After typing ↑C, you may type any commands that do not affect the status of your program (e.g., PJOB, DAYTIME, RESOURCES) and still be able to continue the execution of the program with a CONTINUE command. However, continuing is impossible if you issue any command that runs a new program, such as a RUN or R command.

### 7.5.3 The CONTINUE Command

If you stop your program by a HALT ( ↑C) command, you may resume execution from the point at which it was interrupted by typing the CONTINUE command. You may continue the program only if you exit by typing control - C. If the program exited on an error condition of some sort, the monitor does not let you continue. It types:

CAN'T CONTINUE

if you try. However, you may continue your program if it has halted and given the typeout

HALT AT USER n

## 7.6 COMMANDS TO GET INFORMATION FROM THE SYSTEM.

### 7.6.1 The RESOURCES Command

The RESOURCES command types out a list of all the available devices (except Teletypes) and the number of free blocks on the disk. A disk block is 128 PDP-10 words in length. For example,

\_ RESOURCES )

22100. BLKS,PTYI,CDR,PTR,MTA1,CDP,PLT

At the time of this command, there were 22100 free disk blocks available in addition to six devices.

### 7.6.2 The SYSTAT Command

The SYSTAT command produces a summary of the current status of the system and may be typed without logging in. Included in the summary is a list of the jobs currently logged in, along

with their project-programmer numbers, program names being run, and runtime. The following is a partial example of SYSTAT. More information is contained in this program and can be obtained by running SYSTAT.

UPTIME 01:26:40, 38% NULL TIME = IDLE+LOST = 26% + 12%

| JOB | WHO     | WHERE | WHAT   | SIZE | STATE | RUNTIME  |
|-----|---------|-------|--------|------|-------|----------|
| 1   | **, **  | DET   | PIP    | 1K   | ↑C SW | 00:12:01 |
| 2   | 2,5     | TTY4  | MACRO  | 4K   | ↑C SW | 00:01:22 |
| 3   | 11,554  | TTY3  | MACRO  | 12K  | RN    | 00:05:05 |
| 4   | 40,633  | TTY6  | PIP    | 1K   | TT    | 00:00:32 |
| 5   | 13,575  | TTY20 | COBOL  | 15K  | TT SW | 00:00:19 |
| 6   | 30,637  | TTY24 | DEVCHK | 1K   | TT SW | 00:00:52 |
| 7   | 1,2     | TTY1  | OPFILE | 3K   | IO    | 00:00:20 |
| 8   | **, **  | TTY27 | LOGOUT | 1K   | MQ    | 00:00:24 |
| 9   | 20,521  | TTY13 | SYSTAT | 1K   | ↑C SW | 00:00:58 |
| 10  | 20,623  | TTY23 | PIP    | 1K   | AU    | 00:11:11 |
| 11  | 1,2     | TTY2  | PRINTR | 2K   | IO    | 00:00:39 |
| 12  | **, **  | TTY33 | LOGIN  | 1K   | SL    | 00:00:00 |
| 13  | **, **  | TTY14 | LOGOUT | 1K   | MQ    | 00:02:25 |
| 14  | **, **  | DET   | UFILE  | 2K   | ↑C SW | 00:00:09 |
| 15  | 104,334 | TTY43 | TECO   | 2K   | ↑C SW | 00:00:00 |
| 16  | 70,54   | TTY15 | STACK  | 2K   | TT SW | 00:01:33 |
| 17  | 40,65   | TTY10 | TECO   | 2K   | RN    | 00:02:42 |
| 18  | 40,64   | TTY7  | PRINT  | 1K   | MQ    | 00:00:12 |
| 19  | **, **  | DET   | SYSTAT | 1K   | ↑C SW | 00:00:00 |
| 20  | 11,70   | TTY21 | MONGEN | 3K   | TT    | 00:00:01 |
| 21  | **, **  | DET   | CHKPRT | 0K   | ↑C    | 00:00:01 |
| 22  | 11,131  | TTY37 | HMAC   | 6K   | TT SW | 00:00:42 |
| 23  | 13,12   | TTY5  | BATCH  | 4K   | SL    | 00:00:58 |
| 24  | **, **  | DET   | PLEASE | 1K   | ↑C SW | 00:00:01 |

## 7.7 ADDITIONAL MONITOR COMMANDS

In order to present the complete set of monitor commands, the following must be included. These commands are special run control and system administration commands and are discussed in the PDP-10 Reference Handbook, Communicating With the Monitor, Chapter 2.

Table 7-2  
Additional Monitor Commands

| Command | Explanation                                                                                                                                                                                                               |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DDT     | This command is used in debugging programs and allows DDT to assume control of the execution of your program in various ways. See the PDP-10 Reference Handbook, Communicating with the Monitor, for further information. |



Table 7-2 (Cont.)

| Command         | Explanation                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| REENTER         | Similar to the DDT command. This command causes the program to start at an alternate specified entry point. The alternate entry point must be set by you or your program.                                                                                                                                                                                                                                   |
| E               | This command allows you to examine locations in your core area. The argument adr is required the first time you use this command. By specifying an address, the contents of that location are typed. If an argument is not specified, the contents of the location following the previously specified address are typed out.                                                                                |
| D               | This command allows you to deposit information in your core area. You must specify octal values to be deposited in the left half and right half of the location. The address of the location in which information is to be deposited may be specified. If the address is not specified, the information is deposited in the last location examined or the last location in which information was deposited. |
| SSAVE           | This command is the same as the SAVE command except that the program will be sharable when it is loaded with the GET command. To indicate sharability, the high segment is written with the extension .SHR. A subsequent GET causes the high segment to be sharable.                                                                                                                                        |
| CSTART<br>CCONT | These commands are identical to the START and CONT commands except that you are able to continue talking to the monitor. The START and CONT allow you to communicate only with the CUSP that is in execution. Monitor commands may now be executed while the job is running.                                                                                                                                |
| SCHEDULE n      | This command is legal only from the operator's console. It changes the scheduled use of the system depending on n, where n is:<br>0 = regular time-sharing<br>1 = no further logins allowed<br>2 = no further logins allowed from remote Teletypes                                                                                                                                                          |
| ASSIGN SYS:dev  | This command changes the system device to device "dev". The user must be logged in under [1,1] or [1,2].                                                                                                                                                                                                                                                                                                    |
| DETACH dev      | This command assigns the device "dev" to job 0, thus making it unavailable. The user must be logged in under [1,1].                                                                                                                                                                                                                                                                                         |

Table 7-2 (Cont.)

| Command    | Explanation                                                                                                                  |
|------------|------------------------------------------------------------------------------------------------------------------------------|
| ATTACH dev | This command returns a detached device to the monitor's pool of available resources. The user must be logged in under [1,1]. |
| CTEST      | This command is used to test extensions made to the COMPIL CUSP.                                                             |

## 7.8 MONITOR ERROR MESSAGES

The following table contains a summary of the error messages the system can issue. The convention used in the summary is that

|          |                                                       |
|----------|-------------------------------------------------------|
| dev      | represents any legal device name.                     |
| file.ext | represents any legal filename and filename extension. |
| adr      | represents a user address.                            |
| n        | represents a job number or device unit number         |

Table 7-3

### Monitor Diagnostic Messages

| Message                                  | Meaning                                                                                                                                                                                                                            |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The typein is typed back followed by ? ) | The monitor encountered an incorrect character, such as a letter in a numeric argument. The incorrect character appears immediately before the ? For example,<br><div style="margin-left: 40px;">_ CORE ABC )<br/> CORE A? )</div> |
| ADDRESS CHECK FOR DEVICE dev AT USER adr | The monitor checked a user address and found it to be too large or too small; in other words, the address lies outside the bounds of your program.                                                                                 |
| ALREADY ASSIGNED TO JOB n                | The device is already assigned to another user's job (job n).                                                                                                                                                                      |
| BAD DIRECTORY FOR DEVICE DTAn            | The system cannot read or write the DECTape directory without getting some kind of error. Many times this error occurs when you try to write on a write-locked tape or use a virgin tape.                                          |



Table 7-3 (Cont)

| Message                    | Meaning                                                                                                                                                                                                                                                    |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BUSY                       | The console addressed is not communicating with the monitor or is not positioned at the left margin. (The operator's console is never busy.)                                                                                                               |
| CAN'T ATTACH TO JOB        | The project-programmer number specified is not that of the owner of the desired job.                                                                                                                                                                       |
| CAN'T CONTINUE             | The job was terminated due to a monitor detected error and cannot be continued.                                                                                                                                                                            |
| COMMAND ERROR              | General catch-all error response for the COMPILE commands. The syntax of the command is in error and the command cannot be deciphered.                                                                                                                     |
| dev: ASSIGNED              | The device has been successfully assigned to your job.                                                                                                                                                                                                     |
| DEVICE CAN'T BE REASSIGNED | A user's Teletype cannot be reassigned or an attempt was made to reassign a device that your job is still using.                                                                                                                                           |
| DEVICE dev OK?             | The device is temporarily disabled. The line printer may be turned off or out of paper. For mag tapes, there may be no tape mounted or the switch is in LOCAL. You should correct the situation and then proceed (retry the operation) by typing CONTINUE. |
| DEVICE dev WASN'T ASSIGNED | The device isn't currently assigned to your job and cannot be deassigned or reassigned by your job.                                                                                                                                                        |
| DEVICE NOT AVAILABLE       | Specified device could not be initialized because someone else is using it.                                                                                                                                                                                |
| DIRECTORY FULL             | The directory of the device is full and no more files can be added.                                                                                                                                                                                        |
| DISK NOT AVAILABLE         | The device DSK: could not be initialized.                                                                                                                                                                                                                  |
| ERROR IN JOB n             | A fatal error occurred in your job or in the monitor while servicing your job. This timeout usually precedes a 1-line description of the error.                                                                                                            |
| EXECUTION DELETED          | A program is prevented from being executed because of errors detected during assembly, compilation, or loading. Loading is performed, but the Loader exits to the monitor without starting execution.                                                      |

Table 7-3 (Cont.)

| Message                                      | Meaning                                                                                                                                                             |
|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FILE IN USE OR PROTECTED                     | A temporary command file could not be entered in your UFD (user's file directory).                                                                                  |
| HALT AT USER adr                             | Your program executed a HALT instruction at adr. Typing CONTINUE resumes execution at the effective address of the HALT instruction.                                |
| HUNG DEVICE dev                              | If a device does not respond within a certain period of time when it is referenced, the system decides that the device is not functioning and outputs this message. |
| ILLEGAL DATA MODE FOR DEVICE dev AT USER adr | The data mode specified for a device in your program is illegal, such as dump mode for Teletype.                                                                    |
| ILLEGAL UUO AT USER adr                      | An illegal UUO was executed at user location adr.                                                                                                                   |
| ILL INST. AT USER adr                        | An illegal operation code was encountered in your program.                                                                                                          |
| ILL MEM REF AT USER adr                      | An illegal memory reference was made by your program at adr or adr + 1.                                                                                             |
| INCORRECT RETRIEVAL INFORMATION              | The retrieval pointers for a file are not in the correct format; the file is unreadable.                                                                            |
| INPUT DEVICE dev CANNOT DO OUTPUT            | Output was attempted on a device that can only do input, such as the card reader.                                                                                   |
| INPUT ERROR                                  | I/O error occurred while reading a temporary command file from the disk. File should be rewritten.                                                                  |
| ?INVALID ENTRY - TRY AGAIN                   | An illegal project-programmer number or password was entered and did not match identification in system.                                                            |
| I/O TO UNASSIGNED CHANNEL AT USER adr        | An attempt was made to do an OUTPUT, INPUT, OUT, or IN to a device that your program has not initialized.                                                           |
| JOB NEVER WAS INITIATED                      | An attempt was made to attach to a job that has not been initialized.                                                                                               |
| JOB SAVED                                    | The output is completed.                                                                                                                                            |
| LINKAGE ERROR                                | I/O error occurred while reading a CUSP from device SYS: .                                                                                                          |



Table 7-3 (Cont.)

| Message                                                  | Meaning                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LOGICAL NAME<br>ALREADY IN USE<br>DEVICE dev<br>ASSIGNED | You previously assigned this logical name to another device. The device is assigned but the logical name is not.                                                                                                                                                                                                           |
| LOGIN PLEASE<br>?                                        | A command that requires you to be logged in has been typed to the monitor; it cannot be accepted until you log in.                                                                                                                                                                                                         |
| LOOKUP AND ENTER<br>HAVE DIFFERENT<br>NAMES              | An attempt was made to read and write a file on the disk. However, the LOOKUP and ENTER UO's have specified different names on the same user channel. This message does not indicate a DECTape error.                                                                                                                      |
| MASS STORAGE<br>DEVICE FULL                              | The storage disk is full. Users must delete unneeded files before the system can proceed.                                                                                                                                                                                                                                  |
| NESTING TOO DEEP                                         | The @ construction exceeds a depth of nine and may be due to a loop of @ command files.                                                                                                                                                                                                                                    |
| nK OF CORE<br>NEEDED                                     | There is insufficient free core to load the file.                                                                                                                                                                                                                                                                          |
| n1K BLOCKS OF<br>CORE NEEDED                             | The user's current core allocation is less than the contents of JOBBF.                                                                                                                                                                                                                                                     |
| NON-EX MEM AT<br>USER adr                                | Usually due to an error in the monitor.                                                                                                                                                                                                                                                                                    |
| NON-RECOVERABLE<br>DISC READ ERROR;                      | The monitor encountered an error while reading or writing a critical block in the disk file structure (e.g., the MFD or the SAT table). If this condition persists, the disk must be reloaded using FAILSAFE, after the standard location for the MFD and SAT table has been changed using the monitor once-only dialogue. |
| NON-RECOVERABLE<br>DISC WRITE ERROR;                     |                                                                                                                                                                                                                                                                                                                            |
| NO CORE ASSIGNED                                         | No core was allocated when the GET command was given and no core argument was specified in the GET.                                                                                                                                                                                                                        |
| NO START ADR                                             | Starting address or reenter address is zero because you failed to specify the address.                                                                                                                                                                                                                                     |
| NO SUCH DEVICE                                           | The device name does not exist or all devices of this type are in use.                                                                                                                                                                                                                                                     |

Table 7-3 (Cont.)

| Message                                   | Meaning                                                                                                                                                           |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NO SUCH FILE-<br>file.ext                 | Specified file could not be found. May be a source file or a file required for operation of the COMPILE commands.                                                 |
| NOT A DUMP FILE                           | The file is not a core image file.                                                                                                                                |
| NOT A JOB                                 | The job number is not assigned to any currently running job.                                                                                                      |
| NOT ENOUGH CORE                           | System cannot supply enough core to use as buffers or to read in a CUSP.                                                                                          |
| NOT ENOUGH FREE<br>CORE IN MONITOR        | The monitor has run out of free core for assigning disk data blocks and monitor buffers.                                                                          |
| NOT FOUND                                 | The program file requested could not be found on the system device or the specified device.                                                                       |
| OUT OF BOUNDS                             | The specified adr is not in your core area, or the high segment is write-protected and you do not have privileges to the file which initialized the high segment. |
| OUTPUT DEVICE dev<br>CANNOT DO INPUT      | An attempt was made to input from an output device, such as input from the line printer.                                                                          |
| OUTPUT ERROR                              | I/O error occurred while writing a temporary command file on disk.                                                                                                |
| PC EXCEEDS MEMORY<br>BOUND AT USER adr    | Your program made an illegal transfer outside its bounds.                                                                                                         |
| PROCESSOR CONFLICT                        | Use of + construction has resulted in a mixture of source languages.                                                                                              |
| PLEASE KJOB OR<br>DETACH                  | Attempt was made to attach a job when you already have a job initialized at that Teletype.                                                                        |
| SWAP READ ERROR                           | A consistent checksum error has been encountered when checksumming locations JOBDAC through JOBDAC+74 of the Job Data Area during swapping.                       |
| TOO FEW ARGUMENTS                         | A command has been typed, but necessary arguments are missing.                                                                                                    |
| TOO MANY NAMES<br>or<br>TOO MANY SWITCHES | Command string complexity exceeds table space in the COMPIL CUSP.                                                                                                 |



Table 7-3 (Cont)

| Message                  | Meaning                                                                                                                                                                                                    |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TRANSMISSION<br>ERROR    | During a SAVE, GET, or RUN command, the system received parity errors from the device, or was unable to read your file in some other way. This can be as simple as trying to write on a write-locked tape. |
| TTYn ALREADY<br>ATTACHED | Job number is erroneous and is attached to another console, or another user is attached to the job.                                                                                                        |
| UNRECOGNIZABLE<br>SWITCH | An ambiguous or undefined word followed a slash (/).                                                                                                                                                       |
| UWO AT USER qdr          | This message accompanies many error messages and indicates the location of the UWO that was the last instruction your program executed before the error occurred.                                          |

TO THE HONORABLE  
MEMBERS OF THE  
HOUSE OF REPRESENTATIVES  
IN SENATE CHAMBERS  
WASHINGTON, D. C.  
JANUARY 10, 1945  
SIR:  
I have the honor to acknowledge the receipt of your letter of January 5, 1945, in relation to the above-captioned matter.  
The Bureau is currently reviewing the matter and will advise you of the results of its investigation as soon as possible.  
Very respectfully,  
[Signature]

Very truly yours,  
[Signature]  
[Title]  
[Department]



# **Book 8**

## **Utility Programs**

**BATCH**

**CHAIN**

**LINED**

**TECO**

UNITED

UNITED COMMANDS



## LINED A LINE EDITOR FOR PDP-10 DISK FILES

LINED is a line editor for disk files. It is used to create and edit source program files which are written on disk in ASCII code with line sequence numbers appended. LINED has the ability to reference any line at any time without the user having to close and reopen the file. LINED is a reentrant program and loads in 2K pure and 2K impure segments of core.

### NOTE

In this document, computer typeouts are indicated by underscoring. The symbol ) represents the RETURN key. The symbol \$ represents the ALTMODE key.

## 1. MONITOR COMMANDS

The MONITOR commands CREATE and EDIT may be used to select a file for editing with LINED. A temporary disk file, called ###EDT.TMP, is created for passing the commands to LINED.

### 1.1 The CREATE Command

The CREATE command calls in LINED and opens the specified new disk file for editing. The CREATE command is of the form:

. CREATE filename.ext )

### 1.2 The EDIT Command

The EDIT command calls in LINED and opens the specified existing disk file for editing. The EDIT command is of the form:

. EDIT filename.ext )

## 2. LINED COMMANDS

LINED indicates its readiness to receive commands by typing an asterisk. At this time LINED

is said to be in command mode. The user may then type in the following LINED commands.

## 2.1 Inserting or Replacing a Line

```
* lnnnnn
_
nnnn aaaa.....a
nnnxx ($)
*
_
```

Insert or replace the following typed line at line number nnnnn of the currently open file; nnnnn can be specified as a line sequence number or a point (.), or it can be omitted entirely. A point refers to the last line which was typed, or the last line deleted, or the last line inserted. If nnnnn is omitted, it is assumed to be 10.

When LINED has typed a line sequence number, the program enters text mode. In the text mode, characters typed by the user are understood to be text for the insertion. Following the user's typin of the line to be inserted, LINED types out the next sequential line number (nnnnn+10) following which the user presses the ALTMODE key (sometimes labeled PREFIX or ESC) to terminate the insert process and return to LINED command level.

If there already exists a line at nnnnn, it will be replaced. A single quote following the line number indicates that insertion at this line number will cause the existing line to be replaced.

## 2.2 Inserting Multiple Lines

```
* lnnnnn,iiii
_
nnnnn aaaaa.....a
nnnxx bbbbbb.....b
.
.
nnnyy ($)
*
_
```

Insert the following typed lines, beginning at line number nnnnn (which can be specified as either a line number or a point) of the currently open file. Each time a line is entered, nnnnn is increased by the specified increment, iiii. If iiii is omitted, it is assumed to be 10 (if iiii has never been specified previously), or the previous increment specified.

If nnnnn is omitted, it is assumed to be 10, and the result becomes the line number of the next insertion. Type ALTMODE on the line following the last insertion to return to LINED command mode. LINED then awaits another command.

A double quote following a line number indicates that the increment specified for the current insert instruction has resulted in an existing line being skipped.

## 2.3 Deleting a Line

```
* Dnnnnn
_
```

Delete a line number nnnnn from the currently open file; nnnnn can be specified as either a line sequence number or a point



## 2.4 Deleting Multiple Lines

\*Dmmmm,nnnn

Delete lines mmmmm through nnnnn from the currently open file; mmmmm must be less than nnnnn. Either mmmmm or nnnnn may be specified as a point as long as mmmmm is less than nnnnn.

## 2.5 Printing a Line

\*Pnnnn

Print line nnnnn on the user's Teletype; nnnnn can be specified as either a line sequence number or a point. Typing ALTMODE following a typeout will cause the next sequential line to be printed.

## 2.6 Printing Multiple Lines

\*Pmmmm,nnnn

Print lines mmmmm through nnnnn of the currently open file; mmmmm must be less than nnnnn. Either mmmmm or nnnnn may be specified as a point as long as mmmmm is less than nnnnn.

## 2.7 Closing the Current File

E

Closes the current file and returns to LINED command mode. At this point, the user may either open another file or type  
↑ C to return to Monitor level to assemble, list, and/or save his file on a permanent storage device (e.g., DECtape).

## 2.8 Examples of Command Sequence

### Example 1

.CREATE FILEA

RUN LINED AND OPEN FILE FILEA

\*110

BEGIN INSERTING LINES AT LINE NUMBER  
10 INCREMENTING BY 10.

00010 THE PROGRAM

00020 IS INSERTED

00030 HERE

00350 \$

RETURN CONTROL TO LINED COMMAND  
MODE BY TYPING \$. CLOSE FILE FILEA  
BY TYPING AN E. TYPING A ↑ C RE-  
TURNS TO THE MONITOR COMMAND LEVEL.

\*E

↑ C

:

## Example 2

### EDIT FILE

```
*P10,30
00010 THE PROGRAM
00020 IS INSERTED
00030 HERE
*120
00020 IS PLACED
00030 ($)
*D30
*P10,30
00010 THE PROGRAM
00020 IS PLACED
*E
*↑C
:
```

### RUN LINED AND OPEN EXISTING FILE

```
PRINT LINES 10 THROUGH 30
PRINTOUT
```

```
INSERT LINE 20
```

```
DELETE LINE 30
PRINT LINES 10 THROUGH 30
PRINTOUT
```

```
TYPE E TO CLOSE FILE FILEA
TYPING A ↑C RETURNS JOB TO MONITOR
CONTROL LEVEL.
```

## 3. AUXILIARY COMMANDS

These Auxiliary Commands provide an alternate method of calling LINED and opening files. In most cases, auxiliary commands can be replaced by the monitor instructions CREATE and EDIT (Section 1).

### 3.1 R LINED

LINED can be called in from the system device by typing

```
.R LINED )
```

```
*
```

LINED responds with an asterisk to indicate its readiness to receive a command.

### 3.2 Initializing a File for Processing

```
S filename.ext )
```

Select an existing disk file, filename.ext, for editing.

```
S filename.ext ($)
```

Select (create) a new disk file for editing, calling it filename.ext.

## 4. LINED CONVENTIONS AND RESTRICTIONS

The following conventions and restrictions should be noted:

- Files are written with the installation standard protection. See Book 1 for explanation of protected files.



- b. When in insert mode, typing ALTMODE following the printout of the next insertion line sequence number causes a return to LINED command level. Typing ALTMODE to terminate a line of text to be inserted causes the text line to be ignored.

00010 LINE OF TEXT

00020 (\$) \*

Returns to LINED command level

—

00010

LINE OF TEXT

(\*)

Line is ignored

\*

—

- c. LINED assumes that all blocks in a disk file have an integral number of lines (i.e., each block begins with a sequence number and no line is split between blocks). This will always be the case with files which have been created and edited only with LINED; however, if sequence numbers have been removed, say by TECO, they may be restored by using PIP switch /A (see PDP-10 Reference Handbook).
- d. LINED files can be resequenced using PIP switch /A (see PDP-10 Reference Handbook).
- e. Line number 0 is illegal and cannot be used.
- f. Lines can be edited in any order; however, editing lines by ascending line numbers reduces file access time.

## 5. ERROR HANDLING

When an error is detected, LINED types a message and returns the user to LINED command level (indicated by the output of an \* on the Teletype). Some errors are fatal and cause control to return to the monitor. Error messages for LINED are given in Table 1.

Table 1  
LINED Error Messages

| Message | Meaning                                                                                                                        |
|---------|--------------------------------------------------------------------------------------------------------------------------------|
| ?FAU *  | Filename Already in Use. The filename specified for a newly created file already exists on the disk. Retype command correctly. |
| ?ILC *  | ILlegal Command. Illegal syntax or other error in command string.                                                              |

Table 1 (Cont.)

| Message | Meaning                                                                                          |
|---------|--------------------------------------------------------------------------------------------------|
| ?NCF*   | Not Current File. The filename in an "S filename" command could not be found on the disk.        |
| ?NFO*   | No File Open. No "S filename" command preceded this command string.                              |
| ?NLN*   | Nonexistent Line Number. A Print or Delete command refers to a nonexistent line sequence number. |

## NOTE

The following are internal system errors.

|       |                                                     |
|-------|-----------------------------------------------------|
| ?CCL* | CCL error. Error while referencing CCL comand file. |
| ?COR* | No core available for data segment.                 |
| ?DCE* | Device directory full.                              |
| ?DDE* | Device Data Error. Read or write failure on disk.   |
| ?UNA* | Unit Not Available. The disk is not available.      |

## 6. IMPLEMENTATION

The following explanation is intended to help the user to understand how LINED works so that he may use it more effectively.

Lines of text are stored in a 1000-word working buffer. Each line has a 1-word header containing two items. The left half contains the sequence number of the line, and the right half contains the number of words (including the word containing the line header) needed to store the line of text. Thus, to find the beginning of the next line of text, it is necessary to simply take the address of the current line header and add the word count of the current line.

Several pointer words are used to keep track of the lines in the working buffer. WRTLST contains the sequence number of the highest line in the buffer. SN contains the sequence number of the line currently being handled in a command.

When LINED discovers that SN is greater than WRTLST, it knows that the line being sought has already passed through the working buffer. This line is not directly accessible, because there is no way to read a disk file backwards. Consequently, it is necessary for LINED to close the file and then reopen it. This process of going from the current position of the file to the end of the file, from there to the beginning of the file, and finally to the line being sought is



accomplished as follows:

- a. To close the file, all remaining text must be passed through the working buffer to the temporary output file (called **###LIN.TMP**). This is done by giving the subroutine **FNDLIN** (which finds a line whose sequence number is **SN**) the highest possible sequence number - **99999**.
- b. Next, the original file is renamed to **###TMP.TMP**, the temporary output file is renamed to the original filename and the original file (**###TMP.TMP**) is renamed to **name.BAK** (same name as original with an extension of **BAK**).
- c. **FNDLIN** is then given the sequence number being sought, and **LINED** continues with the original command.

TECO



TECO

TECO



## CHAPTER 1

### INTRODUCTION TO TECO

TECO, a very powerful text editor, enables the advanced PDP-10 user to edit any ASCII text with a minimum of effort. All editing can be accomplished by using only a few simple commands; or the user may select any of a large set of sophisticated commands such as character string searching, command repetition, conditional commands, programmed editing, and text block movement. In this description of TECO only the basic commands are described. If the user requires information about the more advanced uses of TECO, he can refer to the TECO section of the PDP-10 Reference Handbook.

TECO is a character-oriented editor. One or more characters in a line can be modified without retyping the rest of the line. Any sort of document can be edited: programs written in FORTRAN, COBOL, MACRO-10, or any other language; memoranda; specifications; and other types of arbitrarily formatted text. TECO does not require that line numbers or any other extraneous information be associated with the text.

#### 1.1 GENERAL OPERATING PROCEDURE

TECO operates on ASCII data files. A file is an ordered set of data on some peripheral device. In the case of TECO, a data file is some type of document. An input file may be a named file on disk or DECtape, a file on magnetic tape, a deck of punched cards, or a punched paper tape. An output file can be written onto any of these same devices. The input file for a given editing operation is the file to which the user wishes to make changes. If the user is using TECO to create a new file, there is no input file. The output file is either the newly created file or the edited version of the input file. An output file is not required if the user wishes merely to examine a file without making any changes.

In general, the process of editing proceeds as follows. The user first specifies the file he wishes to edit and then reads in a "page" of text. A page is normally an amount of text that is intended for a single sheet of paper. Form feeds are used to separate a document into pages. On input, TECO interprets form feeds as end-of-page indicators. It is not required, however,

that a document be so divided into pages. If a form feed is not encountered, TECO simply reads as much text as will reasonably fit into its editing buffer. For the purposes of this document, the word page is used to mean the segment of text in TECO's editing buffer.

When a page has been read into the buffer, the user can modify this text by using the various editing commands. When he has finished editing the page, he outputs it and reads in the next page. This process continues until, after the last page has been output, the user closes the output file. If there are several pages where no editing is required, there are commands which may be used to skim over them.

## 1.2 INITIALIZATION

The two main uses of TECO are ( 1 ) to create a new disk file, and ( 2 ) to edit an existing disk file. These are the only uses of TECO described in this document. In particular, the use of TECO with devices other than disk is not described. The beginner can get around this limitation by using PIP to transfer files to and from disk. (Refer to Book 6 in the PDP-10 Reference Handbook for information about PIP.)

The two main uses of TECO are so common that there are direct monitor commands to initialize TECO for executing them. The command

```
. MAKE filename.ext )
```

is used to initialize TECO for creating a new disk file. Filename.ext is the name that the user gives to the new file. The filename can be from one to six alphanumeric characters. This is followed (optionally) by a period ( . ) and a filename extension of from one to three alphanumeric characters. The most commonly used filename extensions are:

|      |                              |
|------|------------------------------|
| .F4  | for FORTRAN source programs  |
| .CBL | for COBOL source programs    |
| .MAC | for MACRO-10 source programs |

The MAKE command opens a new disk file to receive output from TECO and gives it the name specified by the user. Once the file has been opened it is then actually created by using the insert and output commands, which are explained in sections 2.5 and 2.6 of this document.

The command

```
. TECO filename.ext )
```



is used to initialize TECO for editing an existing disk file, named filename.ext. The filename and filename extension must be exactly the same as those of the file that is to be edited. The TECO command opens the specified file for input by TECO and opens a new file, with a temporary name, for output of the edited version. When output of the new version is completed, the original version of the file is automatically renamed filename.BAK, and the newly edited version is given the name of the original file. The filename extension .BAK is used for backup files.

After TECO has been initialized for a particular job, it responds by typing an asterisk (\*). The asterisk indicates that TECO is ready to accept commands; it is typed at the beginning of TECO's operation and at the completion of execution of every command string.

Examples:

```

_ MAKE EARNING.F4 )
*
_

```

This command initializes TECO for creation of a new disk file called EARNING.F4. The extension .F4 is used because the file is to be a FORTRAN source file.

```

_ TECO LIB40.MAC )
*
_

```

This command initializes TECO for editing the existing disk file LIB40.MAC. At the completion of editing, TECO automatically changes the name of the original version of LIB40.MAC to LIB40.BAK and gives the name LIB40.MAC to the new version.

#### NOTE

The TECO command cannot be used to edit a file which has the filename extension .BAK. To edit a backup file the user must first rename the backup file. For example, to edit the file LIB40.BAK the user should proceed as follows:

```

_ RENAME LIB40.OLD=LIB40.BAK )
_ TECO LIB40.OLD )
*
_

```

### 1.3 SPECIAL SYMBOLS USED IN THIS DOCUMENT

| Symbol | Character Represented | Comment                                                                                          |
|--------|-----------------------|--------------------------------------------------------------------------------------------------|
| ↵      | Carriage Return       | Whenever the RETURN key is typed, TECO automatically appends a line feed to the carriage return. |
| ⌘      | Altmode               | On most Teletypes, the altmode key is labeled "ALTMODE", but on some                             |

### Special Symbols (Cont.)

| Symbol | Character Represented | Comment                                                                                                                                                                                                                                                                                                                            |
|--------|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ↑C     | Control C             | <p>it is labeled "ESC" or "PREFIX". Since the altmode is a non-printing character, TECO indicates that it has received an altmode type-in by echoing a dollar sign ( \$ ).</p> <p>This character is typed by typing the letter C while holding down the CTRL key. Other control characters are represented in similar fashion.</p> |
| ⓈFORM  | Form Feed             | Form feed is typed by typing ↑F (control F).                                                                                                                                                                                                                                                                                       |
| ↓      | Line Feed             | This symbol is used only when a line feed is explicitly typed. It is not used for the line feed which is automatically assumed when a carriage return is typed.                                                                                                                                                                    |
| →      | Tab                   | Tab is typed by typing ↑I (control I).                                                                                                                                                                                                                                                                                             |
| Δ      | Space                 | This symbol is used occasionally for emphasis.                                                                                                                                                                                                                                                                                     |
| ⓈRO    | Rubout                | This key is used to nullify a character erroneously typed in a command string. Its use is explained fully in Section 1.5.                                                                                                                                                                                                          |

### 1.4 GENERAL COMMAND STRING SYNTAX

TECO commands are usually given by typing the one- or two- letter name of the command. However, many of the commands take arguments. Some typical examples are shown below, to give the reader an idea how TECO commands look. These commands are fully explained later in the manual.

```
L
PW
ISAMPLE Ⓢ
3K
```

TECO commands may be given one at a time. However, it is usually more convenient to type



in a single command string, several commands that form a logical group. An example of a command string is shown below.

\*YIHEADING\$NTAG.\$2LT\$\$

A command string may be typed after TECO indicates its readiness by printing an asterisk. Command strings are formed by merely writing one command after another. Command strings are terminated by typing two consecutive altmodes.

Execution of the command string begins only after the double altmode has been typed. At that point each command in the string is executed in turn, starting at the left. When all commands in the string have been executed, TECO prints another asterisk, indicating its readiness to accept another command.

If some command in the string cannot be executed because of a command error, execution of the command string stops at that point, and an error message is printed. Commands preceding the bad command are executed. The bad command and those following it are not executed.

### 1.5 ERASING COMMANDS

Typographical errors, if discovered while typing a command string, may be "erased" by use of the rubout key. This process is best explained by an example.

\*3LKILEIF ERICXON

After typing this much of the command string, the user discovers that he has misspelled the name "Ericson." To nullify his error, he types three successive rubouts. As he does this, TECO responds by retyping the characters which are being rubbed out.

\*3LKILEIF ERICXON (RO) N (RO) O (RO) X

Of course, rubout is a non-printing character so the actual line looks like this:

\*3LKILEIF ERICXONNOX

Once he has rubbed out the bad character, the user continues the command string from the last correct character.

The actual function of the rubout character is to delete the last typed character in the command string. Consequently, if the bad character is not the last in the string, all characters back to that point must be deleted. Rubout characters do not enter the command string.

An entire command string may be erased, if it has not yet been terminated, by typing two successive ↑G (control G) characters.

Example:

\* 3LKILEIF ERICXON ↑G ↑G

↑G ↑G causes the entire command string to be rejected. TECO types a new asterisk and awaits a new command.

## 1.6 COMMAND ARGUMENTS

There are two types of arguments for TECO commands. Some commands require numeric arguments and some require alphanumeric (text) arguments.

Numeric arguments, and also all numeric type-outs by TECO, are decimal integers. Numeric arguments always precede the command to which they apply. A typical example of a command taking a numeric argument is the command to delete three characters: "3D".

Alphanumeric arguments are textual arguments meant to be interpreted as ASCII code by TECO. Alphanumeric arguments always follow the command to which they apply, and they must always be terminated by an altmode. Examples of alphanumeric arguments are ( 1 ) text to be inserted, and ( 2 ) character strings to be searched for.

Example:

\*ISOMETHING (\$\$)

The argument is "SOMETHING".

As shown in the above example, the altmode used to terminate an alphanumeric argument may also serve as one of the two altmodes necessary to terminate a command string. Any ASCII character except null, altmode, and rubout may be included in an alphanumeric argument.



## TECO COMMANDS

## 2.1 INPUT COMMANDS

The Y (yank) command first clears the editing buffer and then reads the next page of the input file into the buffer.

A single Y command is automatically performed by the command

```
.TECO filename.ext )
```

so that when editing with this command the first page of the input file is automatically read in before TECO prints the first asterisk.

The Y command may be used to delete entire pages of a file, since the editing buffer is completely cleared before the input is performed.

The A (append) command reads in the next page of the input file without clearing the current contents of the editing buffer. This command is used to combine several pages of a document. When the A command is used, the form feed separating the page already in the buffer and the page to be read in is removed. Thus after the A command the two pages are combined into one.

If the editing buffer does not have enough room to accommodate an A command which has been given, TECO automatically expands its buffer and then executes the A command. The user is notified of this action by a message of the following form

```
[ 3K CORE ]
```

If sufficient core is not available to allow buffer expansion, the user is notified by an error message.

### NOTE

On either an A or a Y command the form feed terminating the page to be read in is not actually read into the buffer. It is removed on input and a single form feed is appended to the end of the buffer when the buffer is output.

#### Examples:

.\_ TECO REPORT.CBL )

\*  
-

This command, as part of the process of initializing TECO for editing the disk file REPORT.CBL, automatically clears the buffer and then reads in the first page of the file.

\* Y (\$\$)

\*  
-

This command deletes the entire contents of the buffer and then reads in the next page of the input file.

\* AA (\$\$)

\*  
-

Read the next two pages of the input file into the buffer, combining them with the page already in the buffer.

\* A (\$\$)

[4K CORE]

\*  
-

The buffer is expanded as required by the A command. In most cases this message need be of no concern to the user. It is important only if the system is low on core and does not have swapping capability.

## 2.2 BUFFER POINTER POSITIONING

Since TECO is a character-oriented editor, it is very important that the user understand the concept of the buffer pointer. The position of the buffer pointer determines the effect of many of the editing commands. For example, insertion and deletion always take place at the current position of the buffer pointer.

The buffer pointer is simply a movable position indicator. It is always positioned between two characters in the editing buffer, or before the first character in the buffer, or after the last character in the buffer. It is never positioned "on" a particular character, but rather before or after the character. The pointer may be moved forward or backward over any number of characters.

The J command moves the buffer pointer to the beginning of the buffer, i.e., to the position immediately before the first character in the buffer.

The ZJ command moves the pointer to the end of the buffer, i.e., to the position following



the last character in the buffer.

The C command advances the pointer over one character in the buffer. The C command may be preceded by a (decimal) numeric argument. The command nC moves the pointer forward over n characters. (The pointer cannot be advanced beyond the end of the buffer.)

The R command moves the pointer backward over one character in the buffer. This command may also be preceded by a numeric argument. The command nR moves the pointer backward over n characters. (The pointer cannot be moved backward beyond the beginning of the buffer.)

The L command is used to advance the buffer pointer or move it backward, on a line-by-line basis. The L command takes a numeric argument, which may be positive, negative, or zero, and is understood to be one (1) if omitted.

The action of the L command with various arguments is best explained in a more concrete way. Suppose the buffer pointer is positioned at the beginning of line b, or at some position within line b.

The command L, or 1L, advances the pointer to the beginning of line b+1, i.e., to the position following the line feed which terminates line b.

The command nL, where  $n > 0$ , advances the pointer to the beginning of line b+n.

The command 0L moves the pointer to the beginning of line b. If the pointer is already at the beginning of line b, nothing happens.

The command -L moves the pointer back to the beginning of line b-1.

The command -nL moves the pointer back to the beginning of line b-n.

#### NOTE

After execution of a Y command, the buffer pointer is always positioned before the first character in the buffer. (The Y command automatically executes an implicit J command.) The A command does not change the position of the buffer pointer.

In examples, the position of the buffer pointer is often represented in this manual by the symbol  $\uparrow$  just below the line of text.

#### Examples:

\*J3L (\$\$)

—  
\*  
—

The J command moves the pointer to the beginning of the first line in the buffer. The 3L command then moves it to the beginning of the fourth line.

\*ZJ-2L (\$\$)

—  
\*  
—

This moves the pointer to the beginning of the next to last line in the buffer.

\*L4C (\$\$)

—  
\*  
—

Advance the pointer to the position following the fourth character in the next line.

\*OL2R (\$\$)

—  
\*  
—

OL moves the pointer back to the beginning of the line it is currently on. Then 2R moves it back over the carriage return-line feed pair which terminates the preceding line.

ABCDEF  
↑

In this example of text stored in the buffer, the position of the buffer pointer is shown to be between B and C.

### 2.3 TEXT TYPE-OUT

Various parts of the text in the buffer can be typed out for examination. This is done by use of the T command. Just what is typed out depends on the position of the buffer pointer and the argument given. The T command never moves the buffer pointer.

The command T types out everything from the buffer pointer through the next line feed. Thus, if the pointer is at the beginning of a line, the command T causes that line to be typed out. If the pointer is in the middle of a line, T causes the portion of the line following the pointer to be typed.

The command nT (n > 0) is used to type out n lines, i.e., everything from the buffer pointer through the nth line feed following it.

The command OT types out everything from the beginning of the current line up to the buffer pointer. This is useful for determining the position of the pointer.

The command HT types out the entire contents of the buffer.

The user, especially one new to TECO, should use the T command often, to make sure the buffer pointer is where he thinks it is.



During execution of any T command, the user may stop the Teletype output by typing the ↑O (control O) character. This command causes TECO to finish execution of the command string, omitting all further type-out. The ↑O command does not carry over to the next command string.

Examples:

```
* OLT $$
ENTIRE LINE TYPED
*
```

This command string is used to move the pointer back to the beginning of a line and then type out the entire line. It is frequently used after insertion and search commands.

```
* OTT $$
ENTIRE LINE TYPED
*
```

This command string causes the entire line to be typed without moving the pointer. It is useful after insertion and search commands when it is not convenient to move the pointer back to the beginning of the line.

```
* 2T $$
EF
GHIJKL
*
```

If the buffer contains the text below with the pointer between D and E,

```
ABCDEF )↓
GHIJKL )↓
MNOPQR )↓
```

this command causes the typeout shown.

"ABCD" is not typed because these characters precede the pointer. MNOPQR is not typed because these characters follow the second line feed.

## 2.4 DELETION COMMANDS

Characters are deleted individually by using the D command. The command D deletes the character immediately following the buffer pointer. The command nD, where  $n > 0$ , deletes the n characters immediately following the pointer. The commands -D and -nD delete the character or the n characters, respectively, which immediately precede the buffer pointer.

Lines are deleted using the K command. The K command may be preceded by a numeric argument, which is understood to be 1, if omitted. The command nK ( $n > 0$ ) deletes everything from the current position of the pointer through the nth line-feed character following the pointer. The command HK deletes the entire contents of the buffer.

At the conclusion of a D or K command the buffer pointer is positioned between the characters which precede and follow the deletion.

### Examples:

The editing buffer contains the following three lines of text,  
and the pointer is positioned between the G and H.

ABCDEF<sup>↑</sup>GHIJKLM<sup>↓</sup>

NOPQRSTUVWXYZ<sup>↓</sup>

1234567890<sup>↓</sup>

\* 4D (\$\$)

Delete HIJK.

\*

\* -D (\$\$)

Delete G.

\*

\* -3D (\$\$)

Delete EFG.

\*

\* 7D (\$\$)

Delete HIJKLM<sup>↓</sup> but do not delete the line  
feed at the end of the first line.

\*

\* K (\$\$)

Delete HIJKLM<sup>↓</sup>.

\*

Since the carriage return and line feed at the  
end of the first line are deleted, the text in the  
buffer after this command would be:

ABCDEF<sup>↑</sup>GNOPQRSTUVWXYZ<sup>↓</sup>

1234567890<sup>↓</sup>

\* 2K10D (\$\$)

This would leave the buffer containing only  
ABCDEF<sup>↑</sup>G<sup>↓</sup>.

\*

\* 0LK (\$\$)

This is the command string that is required to  
kill (delete) the entire first line.

\*

\* L2K (\$\$)

This kills the last two lines.

\*

\* HK (\$\$)

Kill the entire buffer.

\*

## 2.5 INSERTION COMMAND

The only insertion command is the I command. The ASCII text that is to be inserted into the buffer is typed immediately after the letter I. The text to be inserted is terminated by an altmode.

Any ASCII character except null, altmode, and rubout may be included in the text to be inserted. Specifically, spaces, tabs, carriage returns, form feeds, line feeds, and control



characters are all allowed. If a carriage return is typed in an insertion, it is automatically followed by a line feed.

The text to be inserted is placed in the buffer at the position of the buffer pointer, i.e., between the characters. At the conclusion of the insertion command the buffer pointer is positioned at the end of the insertion.

Any number of lines may be inserted with a single I command. For the user's protection, however, no more than 10 to 20 lines should be inserted with each I command.

#### Examples:

If the buffer contains ABCDEF↓ with the pointer between D and E, the command

\* IXYZ \$\$ produces ABCDXYZEF↓

\*

-

\* I ) produces ABCD )↓  
\$ \$ EF )↓

\*

-

\* I ↓ produces ABCD ↓  
\$ \$ EF ↓

\*

-

\* 3RI \$ 4CI \$ \$ produces A BCDE F ↓

\*

-

\* (FORM)

\*

-

\$ \$

\*

-

This command is used to separate the page in the buffer into two pages. Both pages, however, remain in the buffer. They are not actually separated until output.

\* JILINE ONE )

LINE TWO )

LINE THREE )

\$ \$

\*

-

This example shows insertion of several lines of text at the beginning of the buffer.

and first line of a document is the  
 \* KI )  
 ( )  
 \* )

This is the command string used to delete the tail of a line without removing the carriage return-line feed at the end of the line. If the buffer contains

```
ABCD )↓
EFGH )↓
```

This command will produce

```
AB )↓
↑EFGH )↓
```

## 2.6 OUTPUT COMMANDS

The command P causes ( 1 ) the entire contents of the editing buffer to be output to the output file and ( 2 ) an implicit Y command to be performed which reads in the next page of the input file. This command is used after editing of a given page is complete and the user is ready to move on to the next page.

The P command may be used with a positive numeric argument to skim over several pages. Specifically, the nP command causes the n consecutive pages of the input file, starting with the page in the editing buffer, to be output, and then the n+1st page to be yanked in.

The PW command merely outputs the page currently in the editing buffer. It does not clear the buffer, it does not read in any more text, and it does not move the buffer pointer. This command is used when creating a new file. It is also used to output the last page of a file.

If the buffer is empty, the PW and P commands have no effect.

The EF command must be used to close the output file after all output to it is complete. EF is normally used after the PW command which outputs the last page of the file.

Examples:

```
* PWEF ( ) ( )
```

This is the command string usually used to close out a file when the last page of the file is in the buffer.

```
* PT ( ) ( )
FIRST LINE
```

This command string outputs the current page, reads in the next page, and then types the first line of the new page.

```
*
-
```



If, for example, page 6 of a document is in the editing buffer, this command causes pages 6 through 13 of the document to be output, one after the other, and then reads in page 14.

## 2.7 SPECIAL EXIT COMMANDS

The EX command is used to conclude an editing job with a minimum of effort. Its use is best shown by an example.

Suppose the user is editing a 30-page file and suppose that the last actual change to the file is made on page 10. At this point the user gives the command

```
* EX @@
```

In this case the action performed by TECO is equivalent to the command string 20PPWEF, with an automatic return to the monitor at the end. Thus, the action of TECO is (1) to rapidly move all the rest of the input file on to the output file, (2) close the output file, and (3) to return control to the monitor.

The EG command is even more efficient. This command performs exactly the same functions as the EX command, but after that it causes re-execution of the last COMPILE, LOAD, EXECUTE, or DEBUG command attempted before TECO was called.

For example, suppose the user gives the command

```
* COMPILE PLOT.F4 )
```

to request compilation of a FORTRAN source program, but the compiler discovers errors in the code. The user would then call TECO to correct these errors:

```
* TECO PLOT.F4 )
```

When all the errors are edited, the user exits from TECO with the command

```
* EG @@
```

This causes the COMPILER command to be executed again on the file PLOT.F4, after TECO has finished output of the file.

Any TECO job may be aborted by using the standard return-to-monitor command: ↑C↑C (control C typed twice). However, if this command is typed before the output file is closed, the output file is lost.

If no input or output operations are in progress a single ↑C is sufficient to exit from TECO to the monitor. In such a case, the user may reenter TECO without destroying the job he was previously executing. This is illustrated in the following example.

```
. TECO SOURCE.MAC) A TECO job is started.
* ICOMMENTS ($$)
* ↑C                               The user exits to perform a few simple monitor
-                                     commands.

. DEASSIGN LPT)
. DAYTIME)
24-FEB-70  10:34
. REE)                             The user reenters TECO. The previous buffer
*                                  is still intact.
-
```

## 2.8 SEARCH COMMANDS

In many cases the simplest way to position the buffer pointer is by using a character string search. A search command causes TECO to scan through the text until a specified string of characters is found, and then to position the pointer at the end of this string. There are two main search commands.

The S command is used to search for a character string within the editing buffer. The string to be searched for is specified as an alphanumerical argument following the S command. This argument must be terminated by an altmode. The character string to be searched for may contain any ASCII character except null, altmode, or rubout.

The S command may be preceded by a numerical argument  $n > 1$ . This argument is used to search for the  $n$ th occurrence of a character string. Thus a 2S command searches for the second occurrence of the particular character string, skipping the first occurrence. If  $n$  is omitted,  $n = 1$  is assumed.



Execution of the S command begins at the position of the buffer pointer and continues to the end of the buffer. If the specified character string is not found in this range, an error message is printed and the buffer pointer is set to the beginning of the buffer.

#### Examples:

```
* SA →B $$
_
*
_
```

This causes the pointer to be positioned after the B in the first occurrence of the string A - tab - B past the current position of the pointer.

```
* J2$NAME $$
_
*
_
```

This causes the pointer to be positioned after the second occurrence of the string "NAME" in the buffer.

```
* S20 )
_
TAG: $ OLT $$
TAG: REST OF LINE
*
_
```

This moves the pointer to the position just following the colon in the string "20) TAG:", then repositions the pointer to the beginning of the line (just before the "TAG:") and types out the entire line starting with "TAG:".

Warning: When attempting a search it is very easy to overlook an occurrence of the search string preceding the one which the user desires. For example, he may want to move the pointer after the word "AND" but erroneously position it after a preceding occurrence of a word like "THOUSAND". For this reason the user, especially the novice, is strongly urged to execute a T command to ascertain the position of the pointer after each search command.

#### Example:

```
* SWORD $ OTT $$
_
FORMAT(1X, 'WORD')
* I△WORD2 $$
_
*
_
```

Here the user wishes to insert "△WORD2" after "WORD". He wisely types out the line to make sure he is at the right place, before inserting "WORD2".

The other principle search command is the N command. The difference is that an S search ends at the end of the current buffer, whereas an N search does not. An N search begins like an S search, but if the character string is not found in the current buffer, an automatic P command is executed. The current page is outputted, the next page read in, and the search continued on the new page. This process continues until either the string is found or the input file is exhausted.

If the N search does find the specified character string, the pointer is positioned at its end.

If the string is not found, an error message is generated. In this case the user caused himself a fair amount of delay. If an N search fails, the user must close the file with an EX command, then reopen it and try the N search again with a character string that can be found. The user is strongly urged to be careful when typing search character strings. Remember also that a search string must be terminated with an altmode.

Example:

\* NSTRING - 3D (\$\$)

? 35

\* EX (\$\$)

. TECO filename.ext )

\* NSTRING (\$- 3D (\$\$)

\*

-

Here the user meant to search for the character string "STRING", and to delete the last three characters of the string. However, he forgot to terminate the search string with an altmode and this caused the unsatisfied search request error message ( ? 35 ).



## CHAPTER 3

### ERROR MESSAGES

When TECO encounters an illegal command or a command that for any other reason cannot be executed, a numeric error message is printed on the user's Teletype. Such messages are of the form ?nn where nn is a two-digit decimal integer that refers to the following table of error messages.

When an error message is generated, the command to which it refers is not executed, the remainder of the command string is ignored, and TECO returns to the idle state by typing an asterisk and awaiting a new command string.

The novice user is especially warned that there are a great many TECO commands that have not been described in this introductory material. Almost every letter of the alphabet and many of the special characters have meanings as TECO commands. Hence, the user should be careful when typing command strings. The beginner should probably stick to relatively short command strings.

In the following table, all TECO error messages are listed, even though some of them refer to the more advanced commands not described in this manual. Error messages referring to the advanced commands will probably be encountered by the user of this introductory material only if he has typed an unintended command letter.

The complete set of TECO commands is fully described in the TECO section of the PDP-10 Reference Handbook. Since most editing can be done using only the basic commands covered in this introductory material, most users should be able to get along without the more advanced description for some time. The novice should gain complete mastery of the basic commands before attempting to use any of the advanced commands.

Table 3-1

## TECO Error Messages

| nn | Meaning                                                                                                                                                                              |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1  | TECO attempted to read commands beyond the terminating <b>\$\$</b> .<br>This error is probably due to an unterminated <b>@</b> or <b>@S</b> , or to an unsatisfied <b>O</b> command. |
| 2  | Error on output device; file closed.                                                                                                                                                 |
| 3  | An attempt was made to supply more than two arguments to a command, either by the use of two commas or by "H,".                                                                      |
| 4  | Too many right parentheses.                                                                                                                                                          |
| 5  | = command with no argument.                                                                                                                                                          |
| 6  | U command with no argument.                                                                                                                                                          |
| 7  | Q, U, X, or G command specifies an illegal Q-register (i.e., other than A through Z or 0 through 9).                                                                                 |
| 8  | In an X command, the second argument is not greater than the first.                                                                                                                  |
| 9  | In a G command, the Q-register does not contain text.                                                                                                                                |
| 10 | In a G command, the data in the Q-register is not in correct form (this is an internal error).                                                                                       |
| 11 | In an Ec command (e.g., ER, EW, EF, etc.), c is illegal.                                                                                                                             |
| 12 | File not found on LOOKUP.                                                                                                                                                            |
| 13 | Blank file name specified for directory device.                                                                                                                                      |
| 14 | Proj-Prog number specified does not have a UFD.                                                                                                                                      |
| 15 | Protection failure on DSK.                                                                                                                                                           |
| 16 | File cannot be accessed because it is currently being written.                                                                                                                       |
| 17 | LOOKUP or ENTER returned error type 6 (not defined).                                                                                                                                 |
| 18 | LOOKUP or ENTER returned error type 7 (no device).                                                                                                                                   |
| 19 | Directory full on ENTER.                                                                                                                                                             |
| 20 | Requested I/O device not available.                                                                                                                                                  |
| 21 | Not assigned.                                                                                                                                                                        |
| 22 | EW command between an EB command and its EF.                                                                                                                                         |
| 23 | EM command given when no input file is open.                                                                                                                                         |
| 24 | nEM command, where n is not in the range from 1 to 16.                                                                                                                               |
| 25 | Internal error: EF and EB, but no input file open.                                                                                                                                   |
| 26 | Illegal character in filename.                                                                                                                                                       |
| 27 | Illegal character in project-programmer number.                                                                                                                                      |
| 28 | Attempt to read an input page when no file has been opened for input.                                                                                                                |



# TECO Error Messages (Cont.)

| nn | Meaning                                                                                                                                    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------|
| 29 | I/O error on input device.                                                                                                                 |
| 30 | Attempt to output a page when no file has been opened for output.                                                                          |
| 31 | Two arguments supplied for an L command.                                                                                                   |
| 32 | Attempt to move the pointer beyond the page.                                                                                               |
| 33 | A 2.-argument command has its second argument less than the first argument.                                                                |
| 34 | Attempt to search for too long a character string.                                                                                         |
| 35 | Search command did not find the requested string.                                                                                          |
| 36 | In an M command, the Q-register does not contain text.                                                                                     |
| 37 | In an M command, the data in the Q-register is not in correct form (this is an internal error).                                            |
| 38 | Unmatched right angle bracket.                                                                                                             |
| 39 | ; encountered when not in an iteration.                                                                                                    |
| 40 | " command with no numeric argument, or "x where x is not a G, L, E, N, or C.                                                               |
| 41 | This is the number typed out at the end of the ? command's dump of the command string in error. Refer to the number of the original error. |
| 42 | A character has been encountered as a command which is not defined.                                                                        |
| 43 | Control D command when DDT is not loaded with TECO.                                                                                        |
| 44 | Not enough core available from the monitor.                                                                                                |
| 45 | Rename command with a name which is blank or one which is already in use. Presumably due to a fault in the EB command.                     |
| 46 | Numeric argument should not be used with EX or EG commands.                                                                                |
| 47 | Using EB command or TECO command with a file having the extension .BAK is illegal.                                                         |
| 48 | ER, EW, EZ, and EB commands may not be used with device TTYn, where TTYn is the user's console or any other attached user's console.       |
| 49 | n <...> where n=0 is illegal.                                                                                                              |

|     |     |
|-----|-----|
| 1   | ... |
| 2   | ... |
| 3   | ... |
| 4   | ... |
| 5   | ... |
| 6   | ... |
| 7   | ... |
| 8   | ... |
| 9   | ... |
| 10  | ... |
| 11  | ... |
| 12  | ... |
| 13  | ... |
| 14  | ... |
| 15  | ... |
| 16  | ... |
| 17  | ... |
| 18  | ... |
| 19  | ... |
| 20  | ... |
| 21  | ... |
| 22  | ... |
| 23  | ... |
| 24  | ... |
| 25  | ... |
| 26  | ... |
| 27  | ... |
| 28  | ... |
| 29  | ... |
| 30  | ... |
| 31  | ... |
| 32  | ... |
| 33  | ... |
| 34  | ... |
| 35  | ... |
| 36  | ... |
| 37  | ... |
| 38  | ... |
| 39  | ... |
| 40  | ... |
| 41  | ... |
| 42  | ... |
| 43  | ... |
| 44  | ... |
| 45  | ... |
| 46  | ... |
| 47  | ... |
| 48  | ... |
| 49  | ... |
| 50  | ... |
| 51  | ... |
| 52  | ... |
| 53  | ... |
| 54  | ... |
| 55  | ... |
| 56  | ... |
| 57  | ... |
| 58  | ... |
| 59  | ... |
| 60  | ... |
| 61  | ... |
| 62  | ... |
| 63  | ... |
| 64  | ... |
| 65  | ... |
| 66  | ... |
| 67  | ... |
| 68  | ... |
| 69  | ... |
| 70  | ... |
| 71  | ... |
| 72  | ... |
| 73  | ... |
| 74  | ... |
| 75  | ... |
| 76  | ... |
| 77  | ... |
| 78  | ... |
| 79  | ... |
| 80  | ... |
| 81  | ... |
| 82  | ... |
| 83  | ... |
| 84  | ... |
| 85  | ... |
| 86  | ... |
| 87  | ... |
| 88  | ... |
| 89  | ... |
| 90  | ... |
| 91  | ... |
| 92  | ... |
| 93  | ... |
| 94  | ... |
| 95  | ... |
| 96  | ... |
| 97  | ... |
| 98  | ... |
| 99  | ... |
| 100 | ... |



# **Appendices**

## **Index**

# popo users bookshelf

## Appendices

## Index

21000

Order No. DEC-10-NGC-00

002 is a trademark and service mark of the RAND Corporation.

A complete guide to the use of the RPD-100 two powerful real-time multiprocessing time-sharing Monitors. All real-time multiprogramming time sharing of the system Monitors is described in this book. The manual is intended to provide facilities to program accept input from and direct output to all system I/O devices and release and protect user programs in storage. This manual is not intended to be used with the Monitors from both a programming and user's viewpoint, and contains several quick-reference tables of commonly used Monitor commands and parameters, as well as a complete listing of the RPD-100 two powerful real-time multiprocessing time-sharing Monitors. All real-time multiprogramming time sharing of the system Monitors is described in this book. The manual is intended to provide facilities to program accept input from and direct output to all system I/O devices and release and protect user programs in storage. This manual is not intended to be used with the Monitors from both a programming and user's viewpoint, and contains several quick-reference tables of commonly used Monitor commands and parameters, as well as a complete listing of the RPD-100 two powerful real-time multiprocessing time-sharing Monitors.



# pdp10 user's bookshelf

## A Bibliography of PDP-10 Programming Documents

APRIL, 1970

To solve several customer problems, the PDP-10 Product Line has initiated a new documentation system. PDP-10 software information is now being printed in two handbooks and a series of notebooks. The handbooks and the notebooks will contain essentially the same material; the notebooks, however, will be updated more frequently with insertable pages.

Two handbooks "PDP-10 Timesharing Handbook" and PDP-10 Reference Handbook" now duplicate software manuals. These handbooks are easy to handle and store, and to assure availability, they are printed in large quantities. Revision and reprinting of the handbooks is done every six months. Customers will receive twenty copies of each handbook with the signing of the purchase order for the PDP-10 and twenty more copies with the delivery of the machine.

Each customer will also receive, free of charge, two copies of the PDP-10 software notebooks—a multiple volume set of manuals in the 8½ by 11" format. The notebooks are printed on high quality paper allowing the customer to effectively reproduce the material. Technical accuracy is maintained with quarterly update pages to the notebooks.

Since most PDP-10 software information is presently included in the handbook, it is no longer necessary to print separate manuals. Therefore, except when information is not contained in the handbooks, individual manuals can no longer be ordered. This Bookshelf serves to indicate in which handbook manuals are now located. If the manuals are not located in either handbook, the order number and price are given.

Available manuals and additional copies of the handbooks may be obtained from Digital Sales Offices or by sending a written request (with check or money order) to Program Library, Digital Equipment Corporation, Maynard.

### PDP-10 Reference Handbook

This handbook is a comprehensive volume of information for experienced programmers, systems analysts, and engineers who are interested in writing and operating assembly-language programs in the PDP-10 time-sharing environment. Included in the handbook are four manuals (System Reference Manual, MACRO-10 Assembler, Time-Sharing Monitors, and DDT-10), editor programs (Editor, LINED and TECO), and utility programs (LOADER, PIP and TENDMP). A sample LOGIN procedure, a convenient summary of monitor commands, and a comprehensive index/glossary make this handbook a valuable reference for the person interested in assembly-language programming. Order No. AIW \$5.00

### PDP-10 Timesharing Handbook

This is a tutorial document intended primarily for students, scientists, engineers, and financial analysts who have no experience in programming. It contains an introduction to timesharing and an explanation of the elementary and advanced monitor commands. Included are the three reference manuals BASIC, AID, and FORTRAN, as well as procedural descriptions of Batch, CHAIN, LINED, and TECO. The four demonstration programs in Book 6 enhance the tutorial aspect of the handbook. Order No. AKW \$5.00

### PDP-10 System Reference Manual

An indexed programmer's handbook that describes the PDP-10 processor and the basic instruction repertoire. Following an introduction to the PDP-10's central processor structure, general word format, memory characteristics, and assembler source-programming conventions, this manual presents the specific instruction format, mnemonic and octal op codes, functions, timing formulas, and examples of each of the basic instructions. Several helpful appendices, including mnemonic op code tables, algorithms and timing charts, complete the manual. Contained in PDP-10 Reference Handbook.

### Time-Sharing Monitors:

A complete guide to the use of the PDP-10's two powerful, real-time, multiprogramming, time-sharing Monitors. All Monitors schedule multiple-user time sharing of the system, allocate facilities to programs, accept input from and direct output to all system I/O devices, and relocate and protect user programs in storage. This manual details user interaction with the Monitors, from both a programming and operating viewpoint, and contains several quick-reference tables of commonly used Monitor commands and parameters, as well as examples of user coding. Contained in PDP-10 Reference Handbook.

### AID (Algebraic Interpretive Dialogue)

A 'hands-on' guide to the use of AID at the Teletype. AID, a PDP-10 version of JOSS<sup>1</sup>, is an on-line system which provides each user with a personal computing service utilizing a conversational algebraic language. This manual describes the use of the Teletype, the syntax and general rules governing the AID language, and each of the AID commands, with appropriate examples. Contained in PDP-10 Time-sharing Handbook.

### Single-User Monitor System

Revision September, 1969

A complete guide to the use of the Single-User Monitor, which performs fast job-to-job sequencing, provides I/O service for all standard devices, and is upward compatible with the Time-Sharing systems.

Order No. DEC-10-MKZA-D

\$2.00

### Batch Processor (Batch) and Job Stacker (Stack)

An indexed manual containing all information required to prepare and run user jobs under control of the Batch Processor in either a single-user or time-sharing environment. (Batch supervises the sequential execution of a series of jobs with a minimum of operator attention, yet allows the operator to interrupt, skip, repeat, or prematurely terminate one or more of the jobs in the series at any time.) Job Stacker is used in conjunction with Batch to (1) transfer job files to the Batch input device and stack them there for subsequent input to Batch, (2) transfer Batch output job files from the Batch output device to some other device, (3) list job file directories, (4) delete job files, and (5) list directories with selective file deletion or transfer. Contained in PDP-10 Time-Sharing Handbook.

### System User's Guide

Revision August, 1969

A fact-filled operations guide designed for handy reference at the user's Teletype. Contains the basics of Teletype usage and complete operating procedures for all Commonly Used Systems Programs (CUSPs). Includes complete write-ups on DECTape Editor, BASIC, LINED, and Linking Loader. A typical chapter includes a brief description of the program, its operating environment, initialization procedures, command string formats, special switches, diagnostic messages, and indepth examples. The manual is tab-indexed for the user's convenience.

Order No. DEC-10-NGCC-D

\$10.00

<sup>1</sup>JOSS is a trademark and service mark of the RAND Corporation for its computer program and services using that program.

## COBOL LANGUAGE

August, 1969

A reference manual designed to aid the user in writing COBOL programs for the PDP-10. Each COBOL language element is accorded a detailed treatment that explains and demonstrates its use in a variety of programming contexts. The four major divisions of a COBOL program and their conventional formats are clearly described and effectively illustrated. Other subjects given extended coverage in this manual are the COBOL library, COBOL reserved words, and the CALL procedure. Each chapter contains numerous examples of the efficient use of the components of a COBOL program. Indexed.

Order No. DEC-10-KC1A-D

\$6.00

## FORTRAN IV

This manual describes statements and features of FORTRAN IV on the PDP-10. Includes descriptions of library functions, calling library subroutines from the Science Library, and the FORTRAN IV operating System. An appendix contains language differences for those using the small (5.5K) PDP-10 FORTRAN Compiler. Contained in PDP-10 Time-Sharing Handbook.

## Science Library and Fortran Utility Subprograms

Revision March, 1969

A general reference manual covering Science Library arithmetic function and utility subprograms and FORTRAN IV nonmathematical utility subprograms. A functional description followed by the calling sequence, list of external subprograms called, entry points, and subprogram length, is given for each subprogram. In addition, the type of argument(s) and result, a description of the algorithm used, and a discussion of the accuracy of the algorithm are given for each function. Appendices contain information on error analyses, double-precision format and input conversion, a bibliography, and average run times.

Order No. DEC-10-SFLE-D

\$4.00

## TECO (Text Editor and Corrector)

Minor Revision, August, 1969

This programmer's reference manual describes the powerful context editor for the PDP-10. Editing is done on a character, line or variable character string basis. Describes more than 30 commands for inserting, deleting, appending, searching for, and displaying text. Contained in PDP-10 Reference Handbook.

## BASIC

A valuable guide to the BASIC® commands needed for a efficient expression of scientific, business, and educational problems. The manual contains complete tutorial explanations of these additional features: (1) matrix computations; (2) alphanumeric information handling; (3) program control and storage facilities; (4) program editing capabilities; (5) formatting of Teletype output; and (6) documentation and debugging aids. Contained in PDP-10 Time-Sharing Handbook.

© Registered: Trustees of Dartmouth College

## PIP (Peripheral Interchange Program)

Explains how PIP is used to transfer data files between standard peripheral devices. Shows how command strings are written, describes switches available for optional functions, techniques for handling file directories, error messages and other features. Contained in PDP-10 Reference Handbook.

## MACRO-10 Assembler

The programmer's reference manual for the PDP-10 assembly system. Explains format of statements, use of pseudo-operations, and coding of macro instructions which make MACRO-10 one of the most powerful assemblers available. Contained in PDP-10 Reference Handbook.

## PDP-10 Reference Card

Revision November, 1969

A handy pocket-sized guide to instruction mnemonics, hardware and software (Monitor system) word formats, and instruction codes.

Order No. DEC-10-J00B-D

\$0.25

## DDT-10 (Dynamic Debugging Technique)

This reference manual describes the dynamic debugging program used for on-line checkout and testing of MACRO-10 and FORTRAN programs. The commands of DDT are grouped so that they can be used easily and effectively by both the uninitiated user and the experienced programmer. Included in the appendices is an informative summary of all DDT functions. Contained in PDP-10 Reference Handbook.

The following supplementary documents are also available from the Program Library.

|                                                                            |                  |      |
|----------------------------------------------------------------------------|------------------|------|
| PDP-10 DECTape Copy Program (COPY)                                         | DEC-10-RPTA-D    | 1.00 |
| FORTRAN IV Software Maintenance Memos                                      | DEC-10-KF1A-D    | 1.00 |
| Linking Loader V.27                                                        | DEC-10-LLZA-D    | 1.00 |
| FORTRAN IV Utility Subprograms (RELEASE, MAGDEN, BUFFER, IFILE, and OFILE) | DEC-10-FIYB-D    | 1.00 |
| S68OI-DC68A Data Line Scanner for PDP-10                                   | DEC-10-FWVA-D    | 1.00 |
| Program Logic Manual for the PDP-10 Time-Sharing Monitors                  | DEC-10-MRZA-D(L) | 4.00 |



# APPENDIX B

## PDP-10 SOFTWARE

Table B-1 lists PDP-10 system programs and the documentation pertaining to them. For a description of each of the documents, refer to the Bookshelf in Appendix A of this handbook.

Table B-1  
PDP-10 Software

| Software | Documentation                                                                           | Document Order No.                    |
|----------|-----------------------------------------------------------------------------------------|---------------------------------------|
| AID      | PDP-10 Timesharing Handbook (Book 4)<br>PDP-10 Software Notebook<br>System User's Guide | AKW<br>DEC-10-SYZA-D<br>DEC-10-NGCC-D |
| BASIC    | PDP-10 Timesharing Handbook (Book 3)<br>PDP-10 Software Notebook                        | AKW<br>DEC-10-SYZA-D                  |
| Batch    | PDP-10 Timesharing Handbook (Book 8)<br>PDP-10 Software Notebook                        | AKW<br>DEC-10-SYZA-D                  |
| BINCOM   | PDP-10 Reference Handbook (Book 6)<br>PDP-10 Software Notebook<br>System User's Guide   | AIW<br>DEC-10-SYZA-D<br>DEC-10-NGCC-D |
| CHAIN    | PDP-10 Timesharing Handbook (Book 8)<br>PDP-10 Software Notebook                        | AKW<br>DEC-10-SYZA-D                  |
| CHKPNT   | PDP-10 Software Notebook<br>(System Manager's Guide)                                    | DEC-10-SYZA-D                         |
| COBOL    | COBOL Language<br>PDP-10 Software Notebook                                              | DEC-10-KC1A-D<br>DEC-10-SYZA-D        |

Table B-1 (Cont.)  
(Cont.)

| Software | Documentation                                                                           | Document Order No.                    |
|----------|-----------------------------------------------------------------------------------------|---------------------------------------|
| COMPIL   | PDP-10 Reference Handbook (Book 2)<br>PDP-10 Software Notebook                          | AIW<br>DEC-10-SYZA-D                  |
| COPY     | PDP-10 DECTape Copy Program (COPY)<br>PDP-10 Software Notebook                          | DEC-10-RPTA-D<br>DEC-10-SYZA-D        |
| CREF     | PDP-10 Reference Handbook (Book 5)<br>PDP-10 Software Notebook                          | AIW<br>DEC-10-SYZA-D                  |
| DDT      | PDP-10 Reference Handbook (Book 5)<br>PDP-10 Software Notebook<br>System User's Guide   | AIW<br>DEC-10-SYZA-D<br>DEC-10-NGCC-D |
| DRIVER   | PDP-10 Timesharing Handbook (Book 8)<br>PDP-10 Software Notebook                        | AKW<br>DEC-10-SYZA-D                  |
| DSKLST   | PDP-10 Software Notebook<br>(System Manager's Guide)                                    | DEC-10-SYZA-D                         |
| EDITOR   | PDP-10 Reference Handbook (Book 4)<br>PDP-10 Software Notebook                          | AIW<br>DEC-10-SYZA-D                  |
| FAILSAFE | PDP-10 Software Notebook<br>(System Manager's Guide)                                    | DEC-10-SYZA-D                         |
| FILDDT   | PDP-10 Software Notebook<br>(System Manager's Guide)                                    | DEC-10-SYZA-D                         |
| FORTRAN  | PDP-10 Timesharing Handbook (Book 5)<br>PDP-10 Software Notebook<br>System User's Guide | AKW<br>DEC-10-SYZA-D<br>DEC-10-NGCC-D |
| FUDGE2   | PDP-10 Reference Handbook (Book 6)<br>PDP-10 Software Notebook                          | AIW<br>DEC-10-SYZA-D                  |



Table B-1 (Cont.)

| Software | Documentation                                                                                          | Document Order No.                    |
|----------|--------------------------------------------------------------------------------------------------------|---------------------------------------|
| GLOB     | PDP-10 Reference Handbook (Book 6)<br>PDP-10 Software Notebook                                         | AIW<br>DEC-10-SYZA-D                  |
| LINED    | PDP-10 Reference Handbook (Book 4)<br>PDP-10 Timesharing Handbook (Book 8)<br>PDP-10 Software Notebook | AIW<br>AKW<br>DEC-10-SYZA-D           |
| LOADER   | PDP-10 Reference Handbook (Book 3)<br>PDP-10 Software Notebook                                         | AIW<br>DEC-10-SYZA-D                  |
| LOGIN    | PDP-10 Reference Handbook (Book 3)<br>PDP-10 Software Notebook<br>(System Manager's Guide)             | AIW<br>DEC-10-SYZA-D                  |
| LOGOUT   | PDP-10 Reference Handbook (Book 3)<br>PDP-10 Software Notebook<br>(System Manager's Guide)             | AIW<br>DEC-10-SYZA-D                  |
| MACRO    | PDP-10 Reference Handbook (Book 2)<br>PDP-10 Software Notebook<br>System User's Guide                  | AIW<br>DEC-10-SYZA-D<br>DEC-10-NGCC-D |
| MONEY    | PDP-10 Software Notebook<br>(System Manager's Guide)                                                   | DEC-10-SYZA-D                         |
| MONGEN   | PDP-10 Software Notebook<br>(System Manager's Guide)                                                   | DEC-10-SYZA-D                         |
| MONITOR  | PDP-10 Reference Handbook (Book 3)<br>PDP-10 Software Notebook                                         | AIW<br>DEC-10-SYZA-D                  |
| PIP      | PDP-10 Reference Handbook (Book 3)<br>PDP-10 Software Notebook                                         | AIW<br>DEC-10-SYZA-D                  |

Table B-1 (Cont.)

| Software | Documentation                                                                                                                 | Document Order No.                           |
|----------|-------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|
| PIP1     | PDP-10 Reference Handbook (Book 3)<br>PDP-10 Software Notebook                                                                | AIW<br>DEC-10-SYZA-D                         |
| PRINT    | PDP-10 Software Notebook<br>(System Manager's Guide)                                                                          | DEC-10-SYZA-D                                |
| PRINTR   | PDP-10 Software Notebook<br>(System Manager's Guide)                                                                          | DEC-10-SYZA-D                                |
| REACT    | PDP-10 Software Notebook<br>(System Manager's Guide)                                                                          | DEC-10-SYZA-D                                |
| SRCCOM   | PDP-10 Reference Handbook (Book 6)<br>PDP-10 Software Notebook<br>System User's Guide                                         | AIW<br>DEC-10-SYZA-D<br>DEC-10-NGCC-D        |
| STACK    | PDP-10 Software Notebook                                                                                                      | DEC-10-SYZA-D                                |
| SYSTAT   | PDP-10 Reference Handbook (Book 3)<br>PDP-10 Software Notebook                                                                | AIW<br>DEC-10-SYZA-D                         |
| TECO     | PDP-10 Reference Handbook (Book 4)<br>PDP-10 Timesharing Handbook (Book 8)<br>System User's Guide<br>PDP-10 Software Notebook | AIW<br>AKW<br>DEC-10-NGCC-D<br>DEC-10-SYZA-D |
| TENDMP   | PDP-10 Reference Handbook (Book 6)<br>PDP-10 Software Notebook                                                                | AIW<br>DEC-10-SYZA-D                         |



## INDEX

Page numbers refer to the book and the page number within the book; i.e., 3-42 is the 42nd page of Book 3.

- A command, TECO, 8-75
- A format, 5-44
- ABS function, 3-14, 3-73
- Absolute value, 3-14, 3-73, 4-10, 4-23
- ACCEPT statement, 5-58, 5-77
- Access time, 1-10
- Access to BASIC, 3-29, 3-32
- Accumulator conventions, 5-111
- Accuracy of AID, 4-11
- Addition, 4-10, 4-23
- Adjustable dimensions, 5-63
- Allocating system resources
  - ASSIGN monitor command, 7-4
  - CORE monitor command, 7-8
  - DEASSIGN monitor command, 7-6
  - FINISH monitor command, 7-7
  - REASSIGN monitor command, 7-7
- Alphanumeric fields, 5-44
- ALPHI. routine, 5-84
- ALPHO. routine, 5-84
- ALTMODE key, 2-20
- Argument, FORTRAN, definition, 5-24
- Arithmetic error conditions,
  - FORTRAN, 5-121
- Arithmetic Formulas, AID, 4-19, 4-78
- Arithmetic function definition
  - statement 5-71, 5-79
- Arithmetic operations, BASIC, 3-13
- Arithmetic operations on complex numbers, FORTRAN, 5-21
- Arithmetic statement, FORTRAN, 5-29
- Array dimensioning, 5-22, 5-62
- Arrays, 4-21, 5-22
- ASCII character set, 5-125
- ASCII mode
  - DECTape, 5-128
  - disk, 5-128
  - magnetic tape, 5-131
- \$Assign Card, Batch control card, 8-20
- PROTECT, 8-20, 8-21
- ASSIGN monitor command, 7-4
- ASSIGN statement, 5-32, 5-77
- ASSIGN SYS monitor command, 7-15
- Assigned GO TO statement, 5-32, 5-77
- Asterisk convention, 2-11
- Asynchronous design, 1-6
- ATN function, 3-14, 3-73
- ATTACH monitor command, 7-8, 7-16
- Augmented matrix, 6-11
- Background job, 7-8
- BACKSPACE statement, 5-39, 5-59, 5-77
- Batch
  - diagnostic messages, 8-28
  - equipment requirements, 8-11
  - examples, 8-33, 8-39
  - introduction, 8-7
  - processing under PDP-10 monitors, 8-7
  - processor, 8-7
- Batch control cards
  - #Tape, 8-21
  - \$\*Card, 8-27
  - \*\*Card, 8-28
  - \$ASSIGN Card, 8-20
  - \$BIN, 8-24
  - \$CBL, 8-21
  - \$CREF, 8-23
  - \$DUMP, 8-27
  - \$EOF, 8-26
  - \$EOJ, 8-26
  - \$EXCDR, 8-23
  - \$F4, 8-21
  - \$GET, 8-25
  - \$JOB, 8-16
  - KEY, 8-18
  - \$LDR, 8-23
  - \$MAC, 8-21

- \$ PAUSE, 8-27
- \$ RUN, 8-25
- \$ SAVE, 8-24
- \$ SSAVE, 8-24
- \$ START, 8-26
- Batch processing, 1-3
- \$ BIN Card, Batch control card, 8-24
- Binary mode
  - DECTape, 5-128
  - disk, 5-128
  - magnetic tape, 5-131
- BINWR. routine, 5-85
- Blank common, 5-64
- Blank fields, 5-49
- Blank records, 5-47
- BLOCK DATA statement, 5-68, 5-76
- BLOCK DATA subprogram, 5-76
- Block identifier, 5-64
- Block name, 5-64
- Block of words, 1-10
- Boolean expressions, 4-11, 4-20, 4-31
- Brackets, AID, 4-10, 4-23
- BTEMP, 8-11, 8-12, 8-19
- Buffer pointer positioning, TECO, 8-76
- Buffer sizes of peripheral devices, 5-127
- BUFFER subroutine, 5-90
  
- C TECO command, 8-77
- CALL statement, 5-75, 5-77
- CANCEL command, 4-37, 4-80
- \$\*Card, Batch control card, 8-27
- \$\$\*Card, Batch control card, 8-28
- Card notation, Batch,
  - #, 8-16
  - braces, 8-16
  - lower case letters, 8-16
  - parameters, 8-16
  - parentheses, 8-16
- Cards acceptable as Batch input
  - card description, 8-16
  - card notation, 8-16
- Carriage control, FORTRAN, 5-45, 5-48
- \$ CBL Card, Batch control card, 8-21
- CCONT monitor command, 7-15
- CHAIN calling sequence, 8-50
- Chain files, 5-90, 8-47
- CHAIN job implementation, 8-47
- CHAIN program description, 8-54
- CHAIN programming considerations, 8-53
- CHAIN subroutine, 5-90, 8-47
- CHANGE instruction, 3-61, 3-63
- Character set, AID, 4-85
- Character set, ASCII, 5-125
- Character set, FORTRAN, 5-17

- Clock, 1-5
- Closed subroutines, 5-71
- COBOL compilation, 8-21
- Coding form, FORTRAN, 5-16
- Command
  - direct, definition, 4-10
  - indirect, definition, 4-10
- Command summary, AID, 4-80
- Comment line, FORTRAN, 5-16
- COMMON area, CHAIN, 8-47
- Common block, FORTRAN, 5-64
- COMMON statement, 5-64, 5-66, 5-78
- Common storage, FORTRAN, 5-64
- COMPILE monitor command, 2-13
- Complex constants, FORTRAN, 5-20
- Complex fields, 5-46
- Complex subexpression, 5-25
- COMPLEX, type declaration
  - statement, 5-68, 5-78
- Compound expressions, FORTRAN
  - logical, 5-28
  - numeric, 5-24
- Compute bound, 1-6
- Computed GO TO statement, 5-31, 5-77
- Conditional expressions, 4-32, 4-78
- Conditional GO TO statement
  - see IF-THEN
- Constants, FORTRAN
  - complex, 5-20
  - double precision, 5-20
  - integer, 5-19
  - literal, 5-21
  - logical, 5-21
  - octal, 5-20
  - real, 5-19
- Context switching, 1-5
- CONTINUE monitor command, 3-32, 4-10, 7-13
- CONTINUE statement, 5-36, 5-77
- Control commands, BASIC, 3-65
- Control statements, FORTRAN, 5-31, 5-77
  - CALL, 5-75
  - CONTINUE, 5-36
  - DO, 5-34
  - END, 5-37
  - GO TO, 5-31
  - IF, 5-32
  - PAUSE, 5-36
  - RETURN, 5-75
  - STOP, 5-37
- CONTROL-C, 2-18, 3-31, 3-66, 4-9
- CONTROL-F, 2-20, 4-11, 4-12
- Core image, 2-3, 2-13
- CORE monitor command, 7-8



- Correcting typing errors
  - AID, 4-13
  - BASIC, 3-30, 3-31, 3-34
  - Monitor, 2-19
  - TECO, 8-73
- Cramer's method, 6-13
- CREATE monitor command, 2-7, 8-59
- Creating files
  - CREATE monitor command, 2-7
  - MAKE monitor command, 2-9
- Creation of CHAIN files, 8-50
- \$CREF card, Batch control card, 8-23
  - cross reference assembly listing, 8-23
- CREF monitor command, 7-10
- Cross-reference listing, 7-10, 8-23
- COS function, 3-14, 3-73
- Cosine, 3-14, 3-73
- COT function, 3-14, 3-73
- Cotangent, 3-14, 3-73
- CSTART monitor command, 7-15
- CTEST monitor command, 7-16
  
- D format, 5-40, 5-44
- D LINED command, 8-61
- D monitor command, 7-15
- D TECO command, 8-79
- Data block, 3-16, 3-61
- Data file capability, BASIC, 3-67, 3-73
- Data line multiplexor, 1-8
- Data line scanner, 1-8
- Data record, 5-53
- Data set, 1-8
- DATA statement, BASIC, 3-11, 3-12, 3-13, 3-16, 3-71
- DATA statement, FORTRAN, 5-66, 5-78
- Data specification statements, 5-61
  - DATA, 5-66, 5-78
  - BLOCK DATA, 5-68, 5-76
- Data specification subprogram, 5-68
- Data transmission statements, 5-39, 5-77
  - ACCEPT, 5-58
  - DECODE, 5-59
  - ENCODE, 5-59
  - PRINT, 5-53
  - PUNCH, 5-54
  - READ, 5-56
  - REREAD, 5-57
  - TYPE, 5-54
  - WRITE, 5-54
- DATA. UUO, 5-85
- DATE subroutine, 5-90
- DAYTIME monitor command, 2-16
- DDT monitor command, 7-14
  
- DDT submode, 8-25
- DEASSIGN monitor command, 7-6
- DEBUG monitor command, 2-14
- Debugging programs
  - DEBUG monitor command, 2-14
  - in Basic, 3-34
- DECODE statement, 5-59, 5-77
- DECTape usage, 5-128
- DEC. UUO, 5-86
- Dedicated system, 1-4
- DEF function, 3-40, 3-74
- Defensive software, 1-12
- DEFINE FILE statement, 5-55
- Defined function, 3-41, 3-74
- Defined locations, 5-111
- DELETE, AID command, 4-39, 4-80
- DELETE, BASIC command, 3-65
- DELETE, monitor command, 2-12
- Deletion commands, TECO, 8-79
- DEMAND command, 4-18, 4-41, 4-80
- Demonstration programs
  - curve-fitting, 6-8
  - random-number generator, 6-36
  - remaindering, 6-26
  - sum of squares, 6-4
- DET function, 3-55, 3-74
- DETACH, monitor command, 7-8, 7-15
- Device assignments, 5-128
- Device control statements, 5-58, 5-77
  - BACK SPACE, 5-59
  - END FILE, 5-59
  - REWIND, 5-59
  - SKIP RECORD, 5-59
  - UNLOAD, 5-59
- Device table, FORTRAN, 5-129
- DEVTB., 5-129
- Diagnostic messages
  - AID, 4-87
  - BASIC, 3-75
  - BATCH, 8-28
  - FORTRAN, 5-116
  - monitor, 7-16
  - LINED, 8-63
  - TECO, 8-87
- Diagnostic software, 1-12
- DIM statement, BASIC, 3-26, 3-28, 3-52, 3-74
- DIMENSION statement, FORTRAN, 5-62, 5-78
  - adjustable dimension, 5-63
- Dimensioning, BASIC, 3-26, 3-28, 3-52, 3-54
- Direct command, definition, 4-10
- Direct steps, 4-15

- Directory format
  - long format, 2-11
  - short format, 7-10
- DIRECTORY, monitor command, 2-11, 7-10
- DIRT. routine, 5-84
- DISCARD command, 4-45, 4-80, 4-84
- Disk usage, 5-128
- Division, 4-10, 4-23
- DO command, 4-18, 4-34, 4-46, 4-80
- DO statement, 5-34, 5-77
- DONE command, 4-51, 4-80
- DOUBLE PRECISION, type declaration
  - statement, 5-68, 5-78
- Double precision constants,
  - FORTRAN, 5-20
- Double word, FORTRAN, 5-25, 5-27
- DOUBT. routine, 5-84
- Driver
  - equipment requirements, 8-30
  - general description, 8-29
  - implementation, 8-30
  - use of, 8-30
- Dual memory protection and
  - relocation, 1-8
- Dummy arguments, AID, 4-20
- Dummy arguments, FORTRAN, 5-72, 5-73
- Dummy identifiers, 5-71, 5-72
- DUMP, 8-27, 8-29, 8-31
- DUMP subroutine, 5-91
- \$DUMP card, Batch control card, 8-27
  
- E format, 5-40, 5-44
- E, LINED command, 8-61
- E, monitor command, 7-15
- Edit commands, BASIC, 3-65
- EDIT, monitor command, 4-9, 8-59
- Editing files
  - EDIT monitor command, 2-9
  - TECO monitor command, 2-10
- EF, TECO command, 8-82
- EG, TECO command, 8-83
- ENC. UUO, 5-86
- ENCODE statement, 5-59
- END OF FILE statement, 5-59
- END statement, BASIC, 3-11, 3-19, 3-72
- END statement, FORTRAN, 5-37
- Entering a BASIC program, 3-30, 3-32
- \$EOF card, Batch control card, 8-26
- EOF1 subroutine, 5-91
- EOF2 subroutine, 5-91
- \$EOJ card, Batch control card, 8-26
- Equipment requirements for BATCH, 8-11
- EQUIVALENCE statement, 5-65, 5-66
  
- Errors, BASIC
  - grammatical, 3-34
  - logical, 3-34
- ERRSET subroutine, 5-91
- EX, TECO command, 8-83
- Examples, BATCH, 8-33, 8-39
- Examples, LINED, 8-61
- EXECUTE, monitor command, 2-14
- Executing a BASIC program, 3-31, 3-33, 3-35
- Executing programs
  - EXECUTE monitor command, 2-14
- Executive mode, 1-9
- Executive program, 1-5
- Exit commands, TECO, 8-83
- EXIT subroutine, 5-91
- \$EXLDR card, Batch control card, 8-23
  - map, 8-23
- EXP function, 3-14, 3-73
- Exponentiation, 4-10, 4-23
- Expression, definition, 4-10, 4-78
- Expressions, FORTRAN
  - logical, 5-26
  - numeric, 5-24
- EXTERNAL statement, 5-76
- External storage, definition, 4-78
- External subprograms, 5-71
  
- F format, 5-40—5-44
- \$F4 card, Batch control card, 8-21
- Field delimiters, 5-43
- Field specifications, 5-40
- Field width, 5-40, 5-44
- FILE command, AID, 4-53, 4-81, 4-84
- File, definition, 4-78
- File protection, 1-11
- Filename extension, 2-6
- Filenames, 2-6
- Files, 2-6
- FILES command, Basic, 3-67, 3-73
- Filing system, 1-10
- FIN. UUO, 5-85
- FINISH, monitor command, 7-7
- FLIRT. routine, 5-84
- FLOUT. routine, 5-84
- FNEND statement, 3-40
- FOR statement, Basic, 3-22, 3-23, 3-71
- FORM command, 4-54, 4-81
- Form, definition, 4-78
- FORMAT statement, 5-39
  - alphanumeric fields, 5-44
  - blank fields, 5-49
  - complex fields, 5-46
  - logical fields, 5-43



mixed fields, 5-45  
 multiple records, 5-46  
 numeric fields, 5-40  
 variable field width, 5-43  
 Formats stored as data, 5-47  
 Formula, 4-19, 4-78  
 FORSE, 5-83  
   FORMAT processing, 5-83  
   I/O device control, 5-84  
   UO dispatching, 5-84  
 FORTRAN compilation, 8-21  
 FORTRAN compiler diagnostics  
   command errors, 5-116  
   compilation errors, 5-117  
 FORTRAN compiler switches, 5-115  
 FORTRAN library functions, 5-86  
 FORTRAN library subprograms, 5-71,  
   5-86, 5-90  
 FORTRAN library subroutines, 5-90  
 FORTRAN operating system, 5-83  
   FORSE, 5-83  
   I/O conversion routines, 5-84  
   FORTRAN UO's, 5-85  
 FORTRAN program and MACRO subprogram  
   linkage, example of, 5-97  
 FORTRAN UO's, 5-85  
 Function, definition, 5-24  
 Function identifier, 5-24, 5-72  
 FUNCTION statement, 5-72  
 FUNCTION subprograms, 5-72  
   FUNCTION statement, 5-72  
 Function subprogram linkage,  
   example of, 5-96  
 Function type, 5-24, 5-73  
 Function value, 5-24  
 Functions, AID  
   argument, 4-26  
   cosine, 4-26  
   digit part, 4-26  
   exponent part, 4-26  
   exponential, 4-26  
   first, 4-26, 4-34  
   fraction part, 4-26  
   integer part, 4-26  
   logarithm, natural, 4-27  
   maximum, 4-27, 4-34  
   minimum, 4-27, 4-34  
   product, 4-27, 4-34  
   signum, 4-27  
   sine, 4-27  
   square root, 4-27  
   sum, 4-27, 4-34  
   truth value, 4-28, 4-32  
 Functions, Basic, 3-14, 3-27, 3-54,  
   3-55, 3-73

G format, 5-40—5-44  
 \$GET card, Batch control card, 8-25  
 GET monitor command, 7-12  
 Getting information from the monitor  
   DAYTIME monitor command, 2-16  
   PJOB monitor command, 2-16  
   RESOURCES monitor command, 7-13  
   SYSTAT monitor command, 7-13  
   TIME monitor command, 2-16  
 Getting on the system, 2-4  
 Glossary of AID terms, 4-78  
 GO command, 4-57, 4-81  
 GO TO statement, Basic, 3-11, 3-18,  
   3-71  
 GO TO statement, FORTRAN  
   assigned, 5-32, 5-77  
   computed, 5-31, 5-77  
   unconditional, 5-31, 5-77  
 GOSUB statement, 3-41, 3-72

H-conversion, 5-45  
 HALT monitor command, 1-10  
 Hierarchy  
   of logical operators, 5-28  
   of numeric operators, 5-26, 5-28  
   of relational operators, 5-28  
 Hilbert matrix, 3-56

I format, 5-40—5-44  
 I LINED command, 8-60  
 I TECO command, 8-80  
 IBUFF call, 5-90  
 Identifiers, 4-10, 4-17, 4-78  
   indexed, 4-21  
 Identity matrix, 3-52, 3-72  
 IF clause, 4-59, 4-81  
 IF END statement, Basic, 3-70, 3-73  
 IF statement, FORTRAN  
   logical, 5-33, 5-77  
   numerical, 5-33, 5-77  
 IF - THEN statement, Basic, 3-10,  
   3-18, 3-71  
 IFILE subroutine, 5-91  
 ILL subroutine, 5-02  
 Immediate storage, definition, 4-78  
 IMPLICIT statement, 5-68, 5-78  
 Impure code, 1-7  
 IN. UO, 5-85  
 Indexed identifiers, 4-21  
 Indirect command, definition, 4-10  
 Indirect steps, 4-15

INPUT file, TECO, 8-69  
 INPUT statement, Basic, 3-48, 3-72  
   data file, 3-69, 3-73  
 Input/output control, 1-9  
 Input/output processor, 1-6  
 Insertion command, TECO, 8-80  
 Instruction set, PDP-10, 5-132  
 INT function, 3-37, 3-73  
 Integer constants, 5-19  
 Integer function, 3-37, 3-73  
 INTEGER, type declaration state-  
   ment, 5-68, 5-79  
 Intelligent terminal, 1-14  
 Interaction, 1-3  
 Internal subprograms, 5-71  
 Interrupting execution of a  
   Basic program, 3-31, 3-35  
 INTL. routine, 5-84  
 INTO. routine, 5-84  
 I/O bound, 1-6  
 I/O conversion routines, 5-84  
 I/O list, 5-52  
 I/O records, 5-53  
 Item-list, definition, 4-78  
 Iterative clauses, 4-34  
  
 J TECO command, 8-76  
 \$JOB card, Batch control card, 8-16  
   DUMP, 8-18  
   NOGO, 8-18  
  
 K TECO command, 8-79  
 KEY card, Batch control card, 8-18  
 KJOB monitor command, 4-17, 4-10  
  
 L format, 5-43  
 L TECO command, 8-77  
 Latency optimization, 1-10  
 \$LDR card, Batch control card, 8-23  
   LIB, 8-23  
   Loader, 8-23  
 Least squares, 6-9  
 Leaving the monitor  
   KJOB monitor command, 2-17,  
   3-32, 3-33  
 LEGAL subroutine, 5-92  
 LENGTH command, 3-65

LET command, AID, 4-19, 4-60, 4-87  
 LET command, Basic, 3-10, 3-16, 3-71  
 LIB40, 5-83  
 LINE command, 4-62, 4-82  
 Line continuation field, 5-15  
 Line format, 5-15  
 Line numbers, Basic, 3-10, 3-12, 3-31  
 Line spacing, 5-48  
 LINED  
   auxiliary commands, 8-62  
   command level, 8-60  
   commands, 8-59  
   conventions and restrictions, 8-62  
   error, 8-63  
   examples, 8-61, 6-62  
   implementation, 8-64  
   monitor commands, 8-59  
   text mode, 8-60  
 LINT. routine, 5-84  
 LIST, Basic command, 3-65  
 LIST, monitor command, 7-9  
 Lists, Basic, 3-25  
 Literal constants, FORTRAN, 5-21  
 LOAD, monitor command, 2-14  
 Loading CHAIN, 8-49  
 Loading programs  
   DEBUG, monitor command, 2-14  
   EXECUTE, monitor command, 2-14  
   LOAD, monitor command, 2-14  
 Locations  
   defined, 5-111  
   required, 5-111  
 LOG, Basic function, 3-14, 3-73  
 LOGICAL, type declaration state-  
   ment, 5-68, 5-79  
 Logical constants, 5-21  
 Logical device names, 7-3  
 Logical expressions, 5-26  
 Logical fields, 5-43  
 Logical IF statement, 5-33, 5-77  
 Logical operators, 5-27, 5-28  
 LOGIN, monitor command, 2-4  
 Login procedure, 2-4  
 Loops, 3-21  
   nested, 3-23  
 LOUT. routine, 5-84  
  
 \$MAC card, 8-21  
 MACRO main programs, 5-103  
 MACRO subprograms, 5-95  
   example, 5-26  
 MAGDEN subroutine, 5-92  
 Magnetic tape usage, 5-130



Magnitude  
   of double precision constants, 5-20  
   of integer constants, 5-19  
   of real constants, 5-19  
 Main memory, 1-5  
 MAKE, monitor command, 2-7, 8-70  
 Manipulating core images  
   GET, monitor command, 7-12  
   R, monitor command, 7-12  
   RUN, monitor command, 7-11  
   SAVE, monitor command, 7-11  
 Manipulating files  
   DELETE, monitor command, 2-12  
   DIRECTORY, monitor command, 2-11  
   RENAME, monitor command, 2-12  
   TYPE, monitor command, 2-11  
 Manipulating Teletype assignment  
   ATTACH, monitor command, 7-8  
   DETACH, monitor command, 7-8  
   TALK, monitor command, 7-8  
 Master File Directory, 1-10  
 Mathematical operators,  
   absolute value, 4-10, 4-23  
   addition, 4-10, 4-23  
   brackets, 4-10, 4-23  
   division, 4-10, 4-23  
   exponentiation, 4-10, 4-23  
   multiplication, 4-10, 4-23  
   parentheses, 4-10, 4-23  
   subtraction, 4-10, 4-23  
 Matrices, 3-51  
 Memory blocks, 1-6  
 Memory protection and relocation,  
   Batch, 8-9  
 Memory protection register, 1-5  
 MFD, 1-10  
 Mixed fields, 5-45  
 Modem, 1-8  
 Modularity, 1-12  
 Monitor, 1-5  
 Monitor calls, 1-10  
 Monitor mode, 1-9  
 Monitor overhead, 1-7  
 MTOP. UUO, 5-85  
 Multiple record formats, 5-46  
   termination of, 5-47  
 Multiplication, 4-10, 4-23  
 Multiprocessor configurations, 1-14  
 Multiprogramming, 1-5  
  
 N, TECO command, 8-85  
 NAMELIST statement, 5-39, 5-49, 5-79  
   input data, 5-50  
   output data, 5-51

Natural logarithm functions, 3-14, 3-73  
 n-dimensional arrays, simulation of, 3-57  
 NEW Basic command, 3-29, 3-65  
 NEXT Basic command, 3-22, 3-23, 3-71  
 NLI. UUO, 5-85  
 NLO. UUO, 5-86  
 NMLST. routine, 5-85  
 Nonexecutable statements  
   FORMAT statement, 5-39  
   NAMELIST statement, 5-49  
 Normal exit of a DO statement, 5-34  
 NUM Basic function, 3-54, 3-74  
 Numbers, Basic, 3-15  
 Numeric expressions, 5-24  
 Numeric fields, 5-40  
   repetition of, 5-46  
   repetition of groups, 5-46  
 Numeric IF statement, 5-33, 5-77  
 Numeric operations, 5-26  
 Numeric operators, 5-24

O format, 5-40—5-44  
 OBUFF call, 5-90  
 Octal constants, 5-20  
 OCTI. routine, 5-85  
 OCTO. routine, 5-85  
 OFILE subroutine, 5-92  
 OLD Basic command, 3-29, 3-65  
 ON-GO TO, Basic, 3-19, 3-72  
 On-line, 1-9  
 Open subroutines, 5-71  
 Operating system, 1-5  
 Operating system diagnostics,  
   FORTRAN, 5-120  
 Operators  
   logical, 5-27  
   numeric, 5-24  
   priorities of, 5-28  
   relational, 5-27  
 Order of precedence, 4-10, 4-23  
 OUT. UUO, 5-85  
 OUTF. UUO, 5-85  
 Output commands, TECO, 8-82  
 Output file, TECO, 8-69  
 Overlapped I/O, 1-10  
 Overlay, 1-8

P LINED command, 8-61  
 P TECO command, 8-82  
 PAGE command, 4-63, 4-82  
 Page, TECO, 8-69  
 Parentheses, 4-10, 4-23

Parts, 4-15, 4-78  
 \$ PAUSE card, Batch control card, 8-27  
 PAUSE statement, FORTRAN, 5-36, 5-77  
 PDUMP subroutine, 5-92  
 Peripheral devices, 7-3  
 Permanent device assignment  
   ASSIGN monitor command, 7-4  
 Permanent resident area (CHAIN), 8-47  
 Physical device names, 7-3  
 PJOB monitor command, 2-16  
 Precision  
   of double-precision constants, 5-20  
   of real constants, 5-19  
 PRINT statement, Basic, 3-11, 3-17,  
   3-45, 3-71  
 PRINT statement, FORTRAN, 5-53, 5-77  
 Printable characters, 3-62  
 Priorities of operators, 5-28  
 Priority interrupt system, 1-5  
 Private device, 1-10  
 Program names, 3-30  
 Programmed operators, 1-10  
 Programming phases, 2-3  
 Propositions, 4-20, 4-31  
 Pseudo-Teletype (PTY), 8-9  
 Public device, 1-10  
 PUNCH statement, 5-54, 5-78  
 Pure code, 1-7  
 PW, TECO command, 8-82

Queue, 1-6  
 QUIT command, 4-64, 4-82

R command, TECO, 8-77  
 R monitor command, 7-12  
   R AID, 4-9  
   R LINED, 8-62  
 Random access of records  
   READ, 5-57, 5-78  
   WRITE, 5-55, 5-78  
 Random number generator, example, 6-36  
 Random numbers, 3-38, 3-73  
 RANDOMIZE statement, 3-39, 3-73  
 Range of a DO statement, 5-34  
 READ statement, Basic, 3-10, 3-11,  
   3-16, 3-71  
 READ statement, FORTRAN, 5-56, 5-78  
 Real constants, 5-19  
 REAL, type declaration statement,  
   5-68, 5-79  
 REASSIGN, monitor command, 7-7

RECALL command, 4-65, 4-82, 4-84  
 Record, 1-10  
 REENTER, monitor command, 4-10, 7-15  
 Re-entrant software, 1-7  
 Relational operators, 5-27  
 Relational symbols, 3-15  
 RELEAS subroutine, 5-92  
 Reliability, 1-12  
 Relocatable binary program, 2-3, 2-13  
 REM statement, 3-49, 3-72  
 Remembered arguments, 2-10  
 Removable resident area (CHAIN), 8-48  
 Removable storage device, 1-11  
 RENAME, Basic command, 3-65  
 RENAME, monitor command, 2-12  
 Repetition  
   of field specifications, 5-46  
   of groups, 5-46  
 REPLACE command, 3-65  
 Replacement operator, 5-29  
 Requesting line printer output  
   CREF, monitor command, 7-10  
   DIRECTORY, monitor command, 7-10  
   LIST, monitor command, 7-9  
 Required locations, 5-111  
 REREAD statement, 5-57, 5-78  
 RERED. UUO, 5-85  
 RESEQUENCE command, 3-66  
 RESET TIMER command, 4-66, 4-82  
 RESET. UUO, 5-85  
 RESOURCES, monitor command, 7-13  
 RESTORE command, 3-49, 3-61, 3-72  
   data file, 3-69, 3-73  
 RETURN key, 2-19  
 RETURN statement, Basic, 3-41, 3-72  
 RETURN statement, FORTRAN, 5-75, 5-77  
 REWIND statement, 5-59, 5-78  
 RND function, 3-38, 3-72  
 Round robin operation, 1-4  
 RTB. UUO, 5-85  
 RUBOUT key, 2-19, 3-30, 3-35  
 Rules of form, AID, 4-11  
 RUN, Basic command, 3-13, 3-31, 3-65  
 \$ RUN card, Batch control card, 8-25  
 RUN, monitor command, 7-11  
 RUNNH command, 3-31, 3-65

S, LINED command, 8-62  
 S, TECO command, 8-84  
 \$SAVE card, Batch control card, 8-24  
 SAVE, monitor command, 3-36, 3-66, 7-11  
 SAVRAN subroutine, 5-92  
 Scalar variables, 5-22



- Scale factor, 5-19, 5-20, 5-42
- SCHEDULE, monitor command, 7-15
- Scheduling algorithm, 1-6
- SCRATCH command, 3-66
  - data file, 3-68, 3-73
- Search commands, TECO, 8-84
- Secondary memory, 1-5
- Segment read-in area (CHAIN), 8-48
- SET command, 4-17, 4-67, 4-82
- SETRAN subroutine, 5-92
- SGN function, 3-40, 3-74
- Shared system device, 8-11
- Sign function, 3-40, 3-74
- SIN function, 3-14, 3-73
- Sine, 3-14, 3-73
- SIZE argument, 4-78
- SKIP RECORD statement, 5-59, 5-78
- SLIST. UUO, 5-85
- SLITE subroutine, 5-93
- Slow peripherals, 1-11
- Source program definition, 2-3
- Spacing, 5-48
- Specification statements, 5-61, 5-78
  - data specification, 5-66
  - storage specification, 5-62
  - type declaration, 5-68
- Spooling, 1-12
- SQR function, 3-14, 3-73
- \$SSAVE card, Batch control card, 8-24
- SSAVE, monitor command, 7-15
- SSWCH subroutine, 5-93
- Standard processor, 2-13
- \$START card, Batch control card, 8-26
- START, monitor command, 7-12
- Starting a core image program
  - CONTINUE, monitor command, 7-13
  - HALT, monitor command, 7-13
  - START, monitor command, 7-12
- Statement field, 5-16
- Statement number field, 5-15
- Statement numbers, 5-15
- Statements, 3-10
- STEP instruction, 3-22, 3-24
- Steps, 4-15, 4-79
- STOP command, AID, 4-68, 4-82
- STOP command, Basic, 3-49, 3-72
- STOP statement, FORTRAN, 5-37, 5-77
- Storage specification statements, 5-62
  - COMMON, 5-64
  - DIMENSION, 5-62
  - EQUIVALENCE, 5-65
- Stored formats, 5-47
- Strings, Basic, 3-59
- Subprogram calling sequences, 5-95

- Subprogram linkage, example, 5-96, 5-104
- SUBROUTINE statement, 5-74
- Subroutine subprograms, 5-74
  - CALL statement, 5-75
  - RETURN statement, 5-75
  - SUBROUTINE statement, 5-74
- Subroutines, 3-41
  - nested, 3-42
- SUBSCRIPT INTEGER, type
  - declaration statement, 5-68, 5-79
- Subscripts, 3-25, 3-28
- Subtraction, 4-10, 4-23
- Swapping, 1-5
- Symbiont operation, 1-12
- Symbolic logic, 5-26
- SYS, 8-11, 8-12
- SYSTAT, monitor command, 7-13
  - example, 7-14
- SYSTEM command, 3-66
- System device, 8-11, 8-12
- T format, 5-48
- T, TECO command, 8-78
- Tab, horizontal, 3-47, 5-15
- Tables, Basic, 3-25
- TALK, monitor command, 7-8
- #TAPE, Batch control card, 8-21
- TAN function, 3-14, 3-73
- TECO command arguments, 8-74
- TECO command string syntax, 8-72
- TECO error messages, 8-87, 8-88
- TECO general operating procedure, 8-69
- TECO initialization, 8-70
- TECO, monitor command, 2-10, 8-70
- TECO special symbols, 8-71
- TECO text type-out, 8-78
- Teletype consoles, 4-12
- Teletype control characters, 2-18
- Teletype special keys
  - ALTMODE key, 2-20
  - RETURN key, 2-19
  - RUBOUT key, 2-19
- Temporary device assignment, 7-4
- Termination of a program, 5-37
- TIME, AID command, 4-79
- TIME, monitor command, 2-16
- Time quantum, 1-4
- Time slice, 1-4
- TIME subroutine, 5-93
- TIMER argument, 4-79
- Timesharing, 1-4
- TO command, 4-69, 4-82

Translating programs  
   COMPILE, monitor command, 2-13  
 Trapping, 1-9  
 TYPE, AID command, 4-71, 4-83  
 Type declaration statements, 5-68  
 TYPE, monitor command, 2-11  
 TYPE statement, FORTRAN, 5-54, 5-78

UFD, 1-10  
 Unconditional GO TO statement, 5-31, 5-77  
 Unit records, 5-40  
 UNLOAD statement, 5-59, 5-78  
 USE command, 4-75, 4-83, 4-84  
 User control levels, Batch  
   console commands, 8-13  
   control cards, 8-13  
 User-defined functions, 4-20, 4-29  
 User directories, 2-11  
 User mode, 1-9  
 User's File Directory, 1-10

Variable field width, 5-43  
 Variables, AID, 4-10, 4-17, 4-79  
 Variables, Basic  
   alphanumeric string, 3-59  
   numeric, 3-15  
   subscripted, 3-25  
 Variables, FORTRAN  
   array, 5-22  
   scalar, 5-22  
 Vectors, 3-51  
 Verbs, AID  
   cancel, 4-37, 4-80  
   delete, 4-39, 4-80  
   demand, 4-18, 4-41, 4-80  
   discard, 4-45, 4-80, 4-84  
   do, 4-18, 4-34, 4-46, 4-80  
   done, 4-51, 4-80  
   file, 4-53, 4-81, 4-84  
   form, 4-54, 4-81  
   go, 4-57, 4-81  
   if clause, 4-59, 4-81  
   let, 4-19, 4-60, 4-81  
   line, 4-62, 4-82  
   page, 4-63, 4-82  
   quit, 4-64, 4-82  
   recall, 4-65, 4-82, 4-84  
   reset timer, 4-66, 4-82  
   set, 4-17, 4-67, 4-82

stop, 4-68, 4-82  
 to, 4-69, 4-82  
 type, 4-71, 4-83  
 use, 4-75, 4-83, 4-84

WEAVE, 3-66  
 Word format, PDP-10, 5-126  
 WRITE statement, Basic, 3-68, 3-73  
 WRITE statement, FORTRAN, 5-54, 5-78  
 WTB. UUO, 5-85

X format, 5-49

Y, TECO command, 8-75



